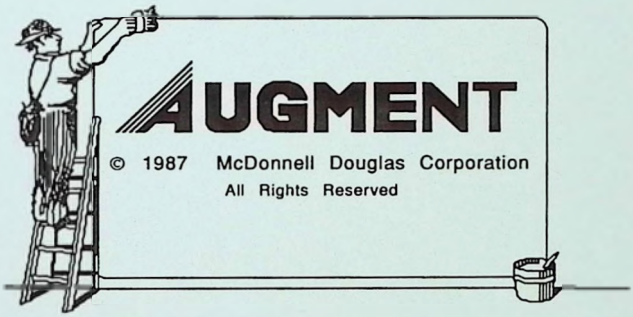
MiniBASE

User's Guide

Version 2.1



Augmentation Systems Division
20705 Valley Green Drive
Cupertino, California 95014

< AUGMENT, 103728, >

TABLE OF CONTENTS

PREFACE

This manual was written to help you master MiniBASE quickly and easily. If you take the time to read through it, and to work through the tutorial exercise included on the MiniBASE disk, you should quickly become adept at using MiniBASE for writing and editing draft documents which can later be easily integrated into AUGMENT.

A manual like this must meet the needs of several kinds of readers. Some of you are already completely familiar with AUGMENT, and will need little introduction to MiniBASE because it is simply a subset of AUGMENT's Base subsystem. You already know how to use it; you simply need to get it installed on a working disk (see the section on "Installation" below) and then quickly familiarize yourself with the differences between MiniBASE and AUGMENT Base.

Others of you are new to AUGMENT but are veteran computer users. You'll need to spend some time learning about AUGMENT itself. You'll find much that's familiar here, but some new concepts too -- for example, AUGMENT uses hierarchically structured files to give you powerful tools for organizing information, and you'll need to become familiar with this new kind of file structure to get the most out of AUGMENT.

And some of you are approaching a computer for the first time; you will find helpful general information in the BACKGROUND section. Readers who know more about using computers may want to skip that section. Note that this introduction does not provide technical details about computer operation or programming.

BACKGROUND

A computer is an electronic device that can accept information from a user, work with the information, and provide results. Computers provide us with an amazing range of tools. Whether you want to solve a math problem, reserve a seat on an airplane, or write a memo, the computer offers a tool that helps you do what you want done. AUGMENT is a set of tools that enables people who work with textual information to use the power of computers. In the case of AUGMENT, it is most important that computers can store text the way filing cabinets or books do and then help you freely copy or change it and deliver it to others. AUGMENT services include access to a computer, equipment that displays information and what you do with it, and the introduction of new skills and working methods to help users take advantage of these tools.

It is worth stopping for a moment to consider what it means to say that computers offer tools. People have been using tools since the days of stone axes; with tools we

have been able to build beyond the strength of our bodies, understand things we cannot observe, and create systems that free us from menial tasks.

Some inventions have opened up the possibility of a wide range of related tools. For example, we speak of the invention of the wheel, but when man began to harness rotation, we were able to have not only wheels, but engines to turn them, mills, drills, watches, and so on. Like rotation, computing is now the basis of many tools. Some very complex tools have been made to feel simple and familiar to the person using them. A car is a complex system with thousands of parts and a very subtle design, yet we can use a car with one steering wheel, some pedals and knobs, and a few weeks training. Moreover, the car is really only part of a more complex system, including roads, gas stations, street signs, and so on, that we employ casually.

People have learned how to apply a computer's power in many ways that have changed the world, but they have only begun to understand how to make a computer simple to use. At McDonnell Douglas' Augmentation Systems Division, we have been working for many years to make computers simple to use. We have succeeded enough to put in your hands more power for many kinds of work than you can get any other way.

Although computers have the potential for many uses, people often adapt them for a narrow purpose, such as producing a certain kind of billing or taking certain kinds of reservations. At the Augmentation Systems Division, we harness computers for the large number of jobs related to working with text. AUGMENT was designed not to solve a specific problem, but rather to help you effectively accomplish many everyday tasks, such as planning and writing documents, exchanging memos and ideas with co-workers, and organizing thoughts and information. In this sense AUGMENT is not one tool itself, but a "system" of coordinated tools like the system that makes highway transportation possible. MiniBASE is one of these tools.

Of course, "everyday" tasks vary from one project to the next, as well as for individual employees, departments, and organizations. Often, you may not even know in advance what kind of help you will need when you are thinking, writing, or trying to find information. AUGMENT is built to accommodate diverse and unexpected needs.

Computers can only respond as they have been designed to respond. In order to get the most from AUGMENT, you must learn what the computer can and cannot do for you, and you must learn the instructions or "commands" to give the computer to make it do what you want. We have designed simple commands to control what the computer does, and there are many such commands to correspond to the many different functions that you need for your everyday work. However, we have

carefully designed the different commands to work in the same way as much as possible so that someone who knows how to use AUGMENT for one job can easily learn what it takes to do another. Once you learn the basics, extending your use of AUGMENT will become like using any of the other familiar tools you employ for your daily work. If, for example, you already are familiar with AUGMENT, you'll find that you already know how to use MiniBASE; conversely, MiniBASE will serve those of you who are new to AUGMENT as an excellent introduction to the system.

INTRODUCTION TO MINIBASE

AUGMENT is a large, powerful set of integrated tools for individual support and for organizational information management. It is a software system which runs in large "mainframe" computers capable of supporting many individual users simultaneously while providing tools far more powerful than can be offered by even the best of today's personal computers. BASE is the AUGMENT subsystem in which most text editing is done. It has commands that allow you to read, write (insert), and modify (edit) online information; print information for offline reading; create, delete, and manage the files in your directory; and control how your terminal displays information.

MiniBASE gives your personal computer a subset of these same AUGMENT Base editing and file handling commands. It turns your personal desktop workstation into a miniature AUGMENT system, allowing you to perform basic text input and editing operations in local or "offline" mode for later transmission to your AUGMENT system, in which higher level editing tools, as well as more specialized information handling tools (e.g., *Mail, Tables,* and *Spell*) are available.

KEYBOARD EQUIVALENCY TABLE

Current Keyboard Assignments

MINIBASE FUNCTION	IBM PC & PC/XT	ZENITH Z-100
OK (UNSHIFTED)	PRTSC/*	LINE FEED
OK	F8	CTRL-D
COMMAND DELETE	DEL	INS LINE
INS	INS	CTRL-E
BACKSPACE CHAR.	BIG LEFT ARROW	BACK SPACE
BACKSPACE WORD	CTRL-W	D CHR
OPTION	F6	CTRL-U

INSTALLATION INSTRUCTIONS

Installing the Mouse (Optional)

The following instructions, which apply to the Mouse Systems Serial Mouse, are an adaptation of the instructions supplied by Mouse Systems for installation of the mouse on an IBM PC or compatible personal computer.

1. If your computer is on, turn it off. Plug one end of the power supply into the jack on the RS-232C connector. Plug the other end into a wall outlet. Look on the underside of your mouse; one (and only one) of the LEDs should be lit. If not, check to verify that the power supply is plugged into a working power outlet and is firmly plugged into the connector.

2. Plug the RS-232C connector from the mouse into the RS-232C port (COM1: or COM2:) on your computer. The cable provided with the mouse should plug securely into the back of the computer without needing an adapter.

3. If you have a modem installed in or connected to your personal computer, it must have two serial ports in order to use MiniBASE with a Mouse Systems Serial Mouse. One port is used for the modem, and the other for the mouse. If you have only one serial port on your personal computer, you will need to purchase another serial port or the Mouse Systems Bus mouse (also available from McDonnell Douglas). If you have two serial ports, but are not sure which one is currently used for the modem, you can use the TEST.COM program supplied on the Mouse Support disk. Place a copy of that disk in a disk drive, type TEST<CR>, and follow the instructions which appear on your screen. Remember that your modem must occupy one serial port and the mouse the other; if, for example, your modem is connected to COM1:, the mouse must be connected to COM2:. The batch files supplied on the Mouse Support Disk assume that the modem is on COM1 and the mouse on COM2; if your personal computer is configured differently you will need to change the batch files.

   Zenith Z-100 users Please Note: The batch files supplied on the Z-100 version of the Mouse Support Disk assume that the modem is on COM2 and the mouse on COM1; if your personal computer is configured differently you will need to change the batch files. In addition, there is no Z-100 version of TEST.COM.

4. Remove and discard the protective plastic covering from the mouse pad. Place the pad in a convenient location near your computer (generally immediately to the right of your keyboard). Orient the pad so that the blue

lines run vertically. (If you view the pad from above, it should be wider than it is tall.) Place the mouse button-side up on the pad.

Note: only one LED on the bottom of the mouse will appear to be lit. The other one is an infra-red LED and will not appear to be on. This is correct and does not indicate a problem with your mouse.

Mouse Support Software

In order to use your mouse, it must be properly connected to your computer as outlined above. This alone, however, will not make the mouse work with MiniBASE; you'll also need to use the mouse driver and menu software included on the disk enclosed with your mouse. To make this easier for you, that disk also includes DOS batch files (extension .BAT) which you can copy to your MiniBASE disk and then use to call MiniBASE. This will ensure that the appropriate mouse support software is loaded each time you run MiniBASE.

The installation procedures described below will allow you the opportunity to install the Mouse Support software along with MiniBASE. If you wish to do so, you should have the Mouse Support Disk available during MiniBASE installation.

If you have already installed MiniBASE and want to install the Mouse Support Software on the same disk, place your MiniBASE disk in one disk drive and the Mouse Support Disk in the other drive and then copy the following files from the Mouse Support disk to the MiniBASE disk:

MOUSESYS.COM	(mouse driver)
M_MB.COM	(custom popup menus for MiniBASE)
MINIBASE.BAT	(batch file to load the mouse driver and the popup menus and then call MiniBASE)

Once you have installed these three files on your MiniBASE disk, whenever you want to run MiniBASE simply type MINIBASE<CR>. You'll be able to see the mouse driver and the menus loaded, after which MiniBASE will load.

Note: if your computer is equipped with a hard disk, you will probably want to install MiniBASE on it as described below (page 7). If you do, you should copy the mouse support software (the files listed above) into the same directory on your hard disk into which you installed the MiniBASE software.

To use the popup menus:

Menus are invoked by pressing the mouse buttons in combination. The two outer buttons together invoke the Main Menu; other combinations may invoke

other menus. The menus you receive may depend on your organization's policy, so just press the buttons and see what's there.

To select an item from a popup menu, simply move the mouse up and down. The selection bar will move vertically within the menu, highlighting each item in turn. When the desired item is highlighted by the selection bar, press the same two buttons again (for the Main Menu, press the two /u}er buttons). The menu will disappear, and the command indicated by the selected menu item will be executed.

To cancel a menu, you can select the 'Exit' item at the bottom of the menu, or you can simply move the mouse left or right out of the menu. In either case the menu will disappear and you'll be back at the MiniBASE command level.

Pressing the mouse buttons one at a time will execute the following MiniBASE command functions:

Command Accept (OK) -- Right Button
Command Delete -- Middle Button
Backspace -- Left Button

Installing MiniBASE

The distribution disk contains all the necessary files to make a working copy of MiniBase. Additionally, if you have compatible mouse software (supplied by McDonnell Douglas on the Mouse Support disk which comes with the mouse), you will be given the opportunity to load the mouse software during this installation. DO NOT use the distribution disk as your working copy.

MAKING A WORKING COPY - FLOPPY DISK SYSTEM

1. Create a bootable disk following the steps outlined in your operating system manual under the FORMAT command. (If you have a Z-100 system and you are using MS-DOS Version 2.x, we recommend that you format the disk selecting the nine sectors per track (/9) option in addition to installing the system.)

2. If you have a Z-100 system, copy the file "ALTCHAR.SYS" from your MS-DOS or Z-DOS distribution disk onto your newly created bootable disk.

3. Place the MiniBASE distribution disk in drive A: and the newly created bootable disk in drive B:

4. Type: A:INSFLOP<RETURN> where <RETURN> is the return key on your computer.

5. The computer should now copy the necessary files from the distribution disk to your bootable disk. During this process, you will have the option of installing mouse software. If you do intend to install the software, depress any key at the appropriate response time. (You will need the Mouse Support disk that came with your mouse to complete this installation.) Otherwise hold the control key down and depress the "C" key. Once the process is complete (indicated by the "Installation complete." message on your screen), remove the distribution disk, insert it in the protective jacket provided and store the disk in a safe place.

MAKING A WORKING COPY - HARD DISK SYSTEM

If your personal computer is equipped with a hard disk drive, you will probably want to store your copy of MiniBASE on the hard disk. This will make the program load and run faster and will generally simplify your use of MiniBASE.

To copy MiniBASE onto your hard disk: We suggest that you create a separate directory on your hard disk to hold all your AUGMENT-related software (Augterm, MiniBASE, and the Mouse Support Software). To do this, first make sure that your computer is on, with the C> prompt (E> for Z-100 users) showing on the screen, and then type MD AUGMENT<CR>. Next, type CD \AUGMENT<CR> to connect to your new AUGMENT directory.

When you have connected to your new AUGMENT directory, place your MiniBASE distribution diskette in drive A: (the floppy disk drive) and type INSHARD<RETURN>. The distribution disk contains the file INSHARD.BAT which copies all the necessary files from the distribution disk to drive C: (drive E: for Z-100 users). Additionally, during this process you will be afforded the opportunity to install MiniBase mouse software. If you do intend to install the software, depress any key at the appropriate response time. If you have no problem with the foregoing then type A:INSHARD.BAT<RETURN>.

If you wish to use another drive, you can either:

1. make the necessary changes to INSHARD.BAT before invoking the file, or

2. manually copy the following files to the appropriate disk/directory.

   *.COM
   *.TXT
   MINIBASE.BAT
   *.AUG

Then manually copy the following files if you wish to load MiniBASE mouse software from the mouse distribution disk.

```
*.COM
*.MBC
MINIBASE.BAT
```

Once you have completed your installation, it is recommended that the distribution disk be placed in the protective sleeve and stored in a save place. It should only be used to restore damaged files.

Loading MiniBASE Automatically: the AUTOEXEC.BAT File

If you use MiniBASE from your IBM PC or PC/XT regularly, you may want to have MiniBASE loaded automatically for you whenever you turn on your computer.

MS-DOS provides a convenient way to do this -- the AUTOEXEC.BAT file. We have included such a file on your distribution disk for your convenience; if you do NOT want MiniBASE loaded for you when you boot the system from this disk, you should either delete this file, or rename it (e.g., RUNMB.BAT). If you rename it or delete it, system startup will request date and time and leave you at DOS command level. Once an "autostart" MiniBASE diskette has been created, turning on the power with this diskette in the left drive will start the MiniBASE program running.

To function properly for "autostart," a system diskette (one formatted with the /S option) should have *at least* the following files copied to it from the MiniBASE diskette (if you have a mouse, you will also need to include the appropriate mouse support software):

```
AUTOEXEC.BAT
MB.COM (current version of MiniBASE)
MB.MBC (MiniBASE configuration file)
WZKB.COM (Z-100 only)
```

If you have created your working copy of MiniBASE by following the instructions above (under "Making a Backup Diskette"), all the necessary files will be present on the disk. The AUTOEXEC.BAT file contains the following instructions to the system:

```
echo off
rem **** COMMAND DISPLAY IS OFF
prompt DOS $N:
WZKB.COM (Z-100 only)
MB MB
```

Note: This file is slightly different in the Z-100 (Z-DOS) version because Z-DOS lacks some of the capabilities used in the MS-DOS version. Also note that in order to use this AUTOEXEC.BAT file on a hard disk you would need to edit it to add a line to connect to the AUGMENT directory before calling MiniBASE.

For more information about the AUTOEXEC.BAT file, or for general information on DOS Batch files, refer to "Batch Processing" in your MS-DOS manual.

TUTORIAL

This section of the manual is designed for people with little or no prior experience with MiniBASE or with AUGMENT. If you are already familiar with either MiniBASE or AUGMENT, you might prefer to skip this section and go to the "REFERENCE" section of the manual, which was written for you.

GETTING STARTED

DISK-BASED TUTORIALS

The best way to learn about MiniBASE is by using it. To help you get off to a fast start, we've included three tutorial files on your MiniBASE diskette; these files can be used in addition to the "Tutorial" section which follows to give you a solid base of MiniBASE skills.

Using the tutorials is simple. Just put your working copy of MiniBASE in drive A, type "tutor<CR>" -- and wait just a few seconds. You'll see the MiniBASE title screen, which will remain for 3-4 seconds; this screen will be followed by a screen full of instructions. Read them, follow them -- and enjoy.

Some suggestions to help you get the most out of the tutorials:

1. Read and follow the instructions carefully. Computers are very literal-minded; it's important that you feed yours exactly the right keystrokes. Strange things may happen otherwise. Also, following the instructions correctly often causes your screen to change -- sometimes these changes are necessary to reveal the next set of instructions.

2. If you want to stop in the middle of a tutorial and come back to it later, you can "freeze" the changes you've made to the file by using the Update command. Just type "ufFILENAME<OK>", where FILENAME is the name under which you'd like to save your working copy of the tutor file you're working on. ("uf" means "Update File"). You can use any name you like, so long as it's no more than eight letters long. Later, when you want to resume work on the tutorial, do NOT type "tutor<CR>" at the DOS prompt; instead, type "mb<CR>" to load MiniBASE, and then type "jfFILENAME<OK>" to load the tutor file you saved earlier.

3. If you think you're getting in trouble and want to wipe out the changes you've made in the tutor file, type "db0<OK>" (for Delete Branch (at) 0). This should leave you with a blank screen. Next, type "jfFILENAME", where FILENAME is the name of the tutor file you were working on, and use the Jump command to move in the file to the location where you had been working.

INTRODUCTION

AUGMENT is a computer system designed to help people work with information. It includes a wide range of tools, from a simple set of commands for reading, writing, and editing documents to sophisticated methods for retrieving and printing information. MiniBASE, a personal-computer based text editor and outline processor, is one of these tools.

Using MiniBASE to write and edit has many advantages over working with paper, pencils, and typewriters. With MiniBASE it is easier to organize your thoughts as well as your documents and to make changes to what you write. Special tools help you collaborate with other people, send messages, prepare correspondence, and so on. Some unique concepts, which you will soon learn, are the basis of new and effective methods of working.

You will begin learning MiniBASE by learning the basic writing and editing commands. Although this is only a small part of what MiniBASE offers, you will be able to see some obvious advantages right away. For example, with a simple command, you can change a document by putting in a comma, taking out a phrase, or adding a paragraph, and MiniBASE will automatically adjust the text so that the document will look as if it had been typed perfectly.

Why use the display terminal and the mouse? When you try to talk to someone about a particular word on a piece of paper, you use either the simple method of pointing to it or the complex method of saying it is seven lines from the top and five words from the left. When you are working in MiniBASE at a standard display terminal, you can use the mouse to point to what you are talking about. If, for example, you want to take out a particular comma on the screen, you can simply give a command and use the mouse to point to the comma you want removed. You can also use the mouse to point to words, pieces of text, paragraphs or headings, and sections of a document.

In MiniBASE, you will always work with material in a "file," a work space on the computer much like a file folder in a filing cabinet. This lesson will teach you how to use the Base subsystem commands to write and edit a simple file containing a business letter.

FILE STRUCTURE

All the information in MiniBASE and AUGMENT files is structured into an outline form. The basic units of information in a file are normally headings or paragraphs, although you are free to choose how to employ the outline structure. As it appears on the screen, indenting shows the structure.

The structure of AUGMENT files helps you organize your thoughts and your documents, and also enables you to locate precisely what you need from a large storehouse of information. The storage area of an AUGMENT host computer contains thousands of books worth of information; this information is divided into directories that belong to people or organizations by name, and into files named by the users. Your personal computer is much smaller, but you'll nevertheless find MiniBASE/AUGMENT file structure a powerful tool for organizing your files and your personal information.

People who are familiar with word processing systems, or with systems based on central computers that address their files by lines, should understand that lines are not a fixed unit in MiniBASE/AUGMENT files. The lines you see are arranged to fit the screen or printed page as the information is on its way from storage to your display or printing device. No AUGMENT file is formatted into pages until it is printed. The fact that lines and pages are not fixed allows a great deal of flexibility when it comes to displaying files or to printing the same files on different devices.

WINDOWS ON THE DISPLAY SCREEN

You start a MiniBASE session from the DOS prompt by typing "mb<CR>." A title screen will be displayed. It will remain on the screen for a short period, and will then disappear, presenting you with an (almost) blank MiniBASE display. (You can make the title screen disappear faster by hitting any key.) You're ready to start work.

The title screen shows (among other things) the version number of MiniBASE you are using. This manual covers version 2.1.2; if your title screen shows a higher version number, please contact the Augmentation Systems Division to obtain the current manual.

When you enter MiniBASE, you will see information displayed on your screen. The screen is actually divided into four areas called "windows." The information displayed in each window has a separate function.

status window (top line of the display): This window displays messages from MiniBASE.

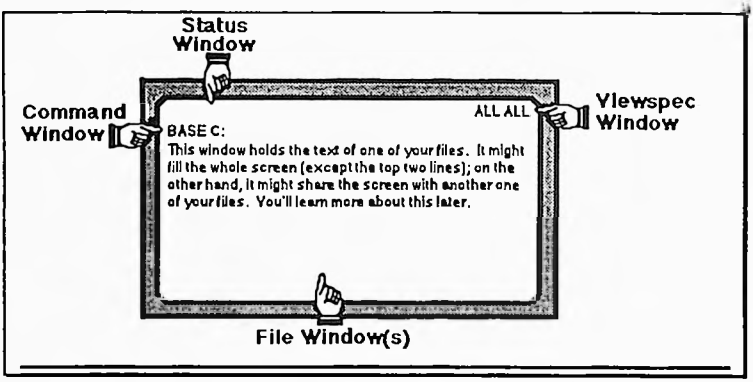
viewspec window (upper right-hand corner): This small window displays information about your view of the file being displayed. For basic information about viewing, see the section on "Writing and Reading an Organized File," which begins on page 35.

command window (second line of the display, including "BASE C:"): As you give a command, it is displayed in this window along with information to prompt

you and help you figure out what to do next. This window also shows the name of the subsystem in which you are working. (This is not applicable in the current release of MiniBASE, but may be used in future releases.)

file window(s): The rest of the display is used for displaying files. Normally, all 22 lines are used to display as much of your current file as will fit on the screen; however, you can subdivide this area into two smaller "windows" as described in the *Reference* section on advanced editing techniques (page 71).

Windows on the MiniBASE Screen



COMMANDS

This section gives you some important general information about MiniBASE commands. It describes features that you will try out later when you begin using the commands introduced in this lesson.

MiniBASE commands use simple English words and were carefully designed to make it easy for you to figure out what you can do next. They all have a similar form, like sentences telling someone to "Do this" or "Put that there." Every command you will learn in this lesson begins with a verb followed by a noun; the verb tells MiniBASE what action to take, and the noun tells it what to act on. For example, the command verb "Delete" will be followed by one of three nouns -- Character, Text, or Word. Understanding the basic form of MiniBASE commands will make it much easier for you to learn new commands and AUGMENT itself.

In addition to having a standard form, MiniBASE commands are entered in a standard way. To give MiniBASE a word in a command, you usually need to type

only the first letter of the word; MiniBASE will recognize the word after the first letter and will display the entire word in the command window. We call a word that MiniBASE recognizes as part of a command a "command word." For example, the letter "d" stands for the command word "Delete", and when you type "d", you see "Delete" in the command window. As you learn MiniBASE, it will help you to pay careful attention to your command window, so you can be sure you are giving the right command.

NOTE: Where MiniBASE will recognize more than one command word beginning with the same letter, you may have to type a space and then one or more letters of the word you want to give. MiniBASE commands are designed, however, so that the most common commands can be entered without typing a space.

Prompts and Noise Words

Before you give a command, or after you have given part of some commands, you will see "C:" in the command window. "C:" is a "prompt." In general, a prompt is one or more uppercase letters, followed by a colon, that tell you what you can do next. "C" in a prompt stands for "command" and means that MiniBASE is waiting for you to enter a command word. For example, after "BASE C:", you may type "d" for "Delete." MiniBASE will recognize the command word after one letter, display the entire word in the command window, and give you a new prompt, "C:", when it is ready for you to do something else.

A slash (/) between the letters in a prompt means that you have a choice. For example, if you type "j" (for "Jump") after "BASE C:," the "C/M:" prompt you see after "Jump" means you can either enter another command word, thus responding to the "C" in the prompt, or "Mark" the location on the screen to which you want to "Jump." The meanings of the letters in MiniBASE prompts will be explained to you as you learn the commands.

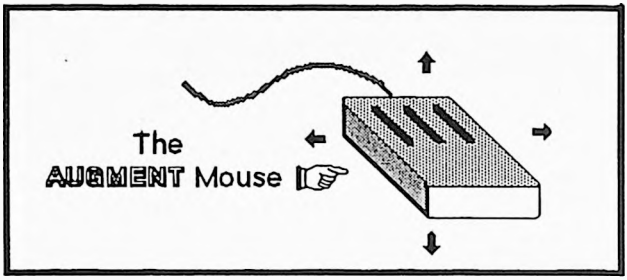
If you type "c" for "Character" in response to the "C:" prompt following "Delete", you see in your command window "Delete Character (at) M/A:". The "at" in parentheses is what we call a "noise word." Noise words provide additional information to help you understand a command. In this case, the "(at)" means you have to think about the location of the character you want to delete.

Marking with the Cursor Keys or the Mouse

To use the basic writing and editing commands in MiniBASE, you must learn how to point and "mark" with the cursor keys or the mouse. Marking is a way of telling MiniBASE exactly what you want a command to act on. You can mark in a MiniBASE command whenever you see "M" in a prompt. For example, when you see the "M/A:" prompt after typing "dc" for "Delete Character", you can mark the

exact character that you want deleted. (The "A" in "M/A" stands for "Address;" you'll learn more about Addresses later.)

If you have a mouse:



Hold the mouse firmly but not tightly with your wrist on the table so you can touch the buttons on top of the mouse with your fingers. Move the mouse across the table and watch the cursor move correspondingly on the screen. When it is appropriate to mark in a command, move the cursor under the character you want to mark and type <OK> by pressing the right mouse button. We show this process as <MARK> in examples. Remember that you must type <OK> after positioning the cursor; moving the cursor without typing <OK> has no effect.

If you do NOT have a mouse:

Use the cursor keys on the right-hand side of your keyboard to move the cursor on the screen. Each time you press one of these "arrow" keys, the cursor will move one character position on the screen. When it is appropriate to mark in a command, move the cursor under the character you want to mark and type <OK> by pressing the key used by MiniBASE as a Command Accept or <OK> key. (On the IBM keyboard this is the "PrtSc/*" or F8 key; on the Z-100, the LINE FEED key. See "Current Keyboard Assignments" on page 3.) We show this process as <MARK> in examples. Remember that you must type <OK> after positioning the cursor; moving the cursor without typing <OK> has no effect.

You can move the cursor faster to the right and left by pressing the control key ("Ctrl") while pressing the arrow keys. This moves the cursor eight character positions at a time. The PgUp and PgDn keys move the cursor vertically about half a screen at a time. (On the Z-100, use the Shift key together with the four arrow keys to move the cursor faster.)

NOTE: Whenever you type <OK>, MiniBASE displays an exclamation point (!) in the command window to let you know it has received the <OK>.

The character you mark will be highlighted. If you accidentally try to mark a position where there is no character, MiniBASE will mark a nearby character for you. It is a good practice to look at your screen when you mark so that you can be sure you have marked the right character.

If you mark the wrong character, you can type <BC> to "backspace a character" and erase the <MARK> by pressing the backspace key or the left mouse button. You can then mark the correct character.You can move the cursor faster to the right and left by pressing the control key ("Ctrl") while pressing the arrow keys. This moves the cursor eight character positions at a time. The PgUp and PgDn keys move the cursor vertically about half a screen at a time. (On the Z-100, use the Shift key together with the four arrow keys to move the cursor faster.)You can move the cursor faster to the right and left by pressing the control key ("Ctrl") while pressing the arrow keys. This moves the cursor eight character positions at a time. The PgUp and PgDn keys move the cursor vertically about half a screen at a time. (On the Z-100, use the Shift key together with the four arrow keys to move the cursor faster.)

Confirming Commands with <OK>

In MiniBASE commands, you type <OK> not only to mark but also to complete other parts of a command and to indicate the end of the command. For example, when you type text in a command, you indicate that you are finished by typing <OK>. If there are no more steps in the command, this <OK> also tells MiniBASE that you are finished with the command. If you mark something as the last step, MiniBASE will prompt you to type a final <OK> with an "OK:" prompt, so you can erase the <MARK> if you wish before ending the command.

You can type <OK> by pressing either the right mouse button, as recommended for marking, or the key for <OK> on the keyboard.

After the final <OK>, MiniBASE will carry out the command as entered, erase it from the command window, and prompt you with "BASE C:" to begin another command.

Cancelling a Command

After you begin a command, you may change your mind or realize you have made a typing error and started a command you do not want. To cancel a command, type the special "command delete" character, <CD>. When you type <CD>, the command you were giving will disappear and the command window will display

only the first prompt, "BASE C:", telling you that MiniBASE is again ready for you to begin a command. For example, this is how you would cancel the Delete Character command:

You type :	Command window shows :
d	Delete C:
c	Delete Character (at) M/A:
<CD>	BASE C:

If you want to erase not the entire command but only the last step in the command, use <BC>.

WRITING: THE INSERT STATEMENT COMMAND

When you first enter MiniBASE you are presented with a blank space in which to begin work. Later on you'll learn how to load files you've saved in earlier work sessions, but for now, let's begin with this fresh work space, which you can think of as an empty, unnamed file. (Actually, it's not a file yet -- but it will be soon.)

When you want to put information into a new MiniBASE file, you add it in the form of statements at the top of the file. The statement is the basic unit of information in MiniBASE and can be anything you feel is a logical unit. For example, if you were using MiniBASE to make a list, you might want to make each item in the list a separate statement. Or if you were writing a letter in MiniBASE, the various parts of the letter -- the salutation, each paragraph in the body, and the closing -- would all be different statements.

Although you can put anything you want in any statement, the first statement in a file -- the one at the very top -- is special in both AUGMENT and MiniBASE. It's called the "origin statement," and it usually looks very much like this:

<MYFILE.AUG,>, 25-Sep-85 16:20 ;;;;

The origin statement contains a lot of information about the file, and it is useful to be able to read it. (AUGMENT users have probably already noticed that this is simply a shorter version of the AUGMENT origin statement.)

The first word in the origin statement is the name of your file. Then there is a period, followed by "AUG". This is called a file extension, and identifies the type of file. In MiniBASE, almost all of the files you will deal with have the "AUG" extension (files created by other personal computer software packages may have other extensions, like "TXT," "DOC," or "DIF"). Don't worry about this now.

All of the above information is contained within angle brackets, and outside those angle brackets is the date and time that the file was last updated. In the case of a

newly created file, it is simply the date and time of creation. The four semicolons at the end of the origin statement mark the end of the "official," system-maintained origin statement; you are free to add any text you like after these semicolons, but should never edit any of the text preceding them.

The command to create an origin statement, and thus to begin a new file, is simply "Create File." You will need to type a leading space because typing simply "c" gives you the Copy command, which will be discussed later. After you have typed "<SP>crf", you will be prompted for the name for the new file. If you type this, followed by <OK>, MiniBASE will insert a new origin statement at the top of the screen.

You only need to use the Create File command once for each file, just to get the origin statement inserted in the proper format. After that, of course, you'll want to get down to the business of writing -- that is, you'll need to start putting information into your file. To enter statements into a file, use the *Insert Statement* command.

If you're reading this tutorial near your personal computer, we suggest that you try these commands, using the following sample business letter for practice.

```
Dear Ms. Jones:

I would like to take this opportunity to thank you for
your presentation last week.  The people present were
impressed with your demonstration, and you certainly
indeed presented us with some interesting alternatives.
I am currently planning to evaluate several products.
Our staff will later this week be meeting to discuss a
purchase.  I will be in touch with you Tuesday of next
week to work out the involved details.

Sincerely Yours,
Elinore Botoh
```

Once you've entered MiniBASE and see the "BASE C:" prompt on the screen, you'll need to use the Create File command as described above to insert an origin statement in your file. Type "<SP>crfletter<OK>." An origin statement will be inserted at the top of your screen; it will look like this (but the date and time will be different):

```
<LETTER.AUG,>, 25-Sep-85 16:45 ;;;;
```

Now you're ready to start typing the letter, each section of which should be entered as a separate statement. Begin the Insert Statement command by typing "is". In the command window, you will see "Insert Statement (at) M/A:". You must then indicate which statement you want your new statement to follow; in this case, you should mark the origin statement (you really haven't much choice at this point,

anyway -- where else could you put it?). Your next prompt will be "L:". When you see this prompt, respond by typing <OK> (we'll explain more about the "L:" prompt later), and then typing the salutation, "Dear Ms. Jones:", and ending it with <OK>. Entering your first statement would look like this:

You type	Command window shows
i	Insert C:
s	Insert Statement (at) M/A:
<MARK>	Insert Statement (at) I L:
<OK>	Insert Statement (at) I I M/T:
Dear Ms. Jones:<OK>	Insert Statement (at) I I Dear Ms. Jones:I BASE C:

When you type the <OK> at the end of the command, the statement you have been typing in the command window will be inserted in place in the file. After you enter the first statement, you will see it in your file window at the top of the screen, just after the origin statement.

For the second statement, type "is" for "Insert Statement" and mark one of the characters in the statement you just added, so the body of the letter will follow the salutation. When typing this long paragraph, you do not have to type a return to indicate the end of a line. Simply type the words, spacing, and punctuation that you want. As you type, you will see the text you're entering appear at the top of the screen. When you reach the right edge of the display, just keep on typing; MiniBASE will automatically adjust the text, starting a new line when it cannot fit any more text on the current line (because it has reached the right edge of the command window). When breaking the text into lines, MiniBASE breaks between words. The text is always adjusted in this way as soon as it is shown in the file window.

When you enter a lot of text, it may seem that what you are typing is writing over the top lines of your file. What is actually happening is that, in order to display the text you are typing, the command window is expanding into the file window. When you end the command and the text is inserted, the command window will return to its former size and your file will reappear.

Try to type the paragraph exactly as it is shown; later in this lesson you will learn how to use editing commands to correct it. If you make mistakes while typing, you can use <BC> to erase the last character you typed and <BW>, the "backspace word" character, to erase the last word plus any spaces or punctuation that immediately follow it. If you type <BC> or <BW> several times, the last several characters or words you typed will be erased. Type <OK> when you have finished the paragraph.

You type	: Command window shows
i	Insert C:
s	Insert Statement (at) M/A:
<MARK>	Insert Statement (at) ! L:
<OK>	Insert Statement (at) ! M/T:
I would like to....	I would like to....
.... details.<OK>	.... details. !
	BASE C:

Complete the letter by adding the last two statements (the closing and the name). For each one, use the Insert Statement command and mark the statement that you want the new statement to follow. If you accidentally type the wrong command, use <CD> to cancel the command, and then start over.

EDITING WITH DELETE, MOVE, REPLACE, AND INSERT

After you enter information in a MiniBASE file, you may want to correct errors or make changes. Three important command verbs for editing text are Delete, Move, and Replace. In addition, the Insert verb, which you have learned to use to add statements to a file, also lets you add text within statements.

Delete enables you to remove information from a file. For example, if you deleted the second "r" in "perrfect", the word would be "perfect".

Move allows you to reorder information in a file by taking it from one place and putting it in another place. For example, if you moved the word "that" in "a phrase makes that sense" to follow the word "phrase", you would have "a phrase that makes sense".

Replace lets you remove information and put other information in its place; that is, it combines Delete and Insert into one verb. The new information may be shorter or longer than what it replaces. For example, if you replaced the word "good" with the word "terrific" in the phrase "a good sentence", you would have "a terrific sentence".

After you edit with any of these verbs, MiniBASE adjusts the text to reflect the change. For example, when you delete information in a statement showing on your

screen, MiniBASE shows the statement as if the information had never been there, possibly breaking the text into lines differently from the way it did before.

Once you know the command verbs, you need to know the nouns that go with them, to tell MiniBASE what you want to delete, move, replace, or insert. Three of the most important nouns are Character, Text, and Word.

A ***Character*** is a single letter, number, punctuation mark, space, return character, or special control character. Characters you can see on your screen are called *visible* characters; those you cannot see are *invisible*. Invisible characters can be marked and edited just like any other characters in MiniBASE statements. A space, for example, is an actual character that separates one word from another and can be deleted, moved, replaced, or inserted; it is not just emptiness.

Text is any series of characters within a single statement. It may begin or end within a word and may include punctuation, spaces, and any other visible or invisible characters. Text can be only one character or it can be all the characters in a statement. You point to text by marking the beginning character and the ending character.

A ***Word*** is a series of letters and/or numbers surrounded by spaces, punctuation marks, or any other characters that are not letters or numbers. (It does not have to be spelled correctly or mean something in English or any other language!) MiniBASE does not consider the surrounding characters as part of the word. You point to a word by marking any character within it. Whenever you delete, move, replace, or insert a word, you do not need to worry about the spacing around the word; MiniBASE knows words commonly have spaces around them and will provide and remove them as necessary.

To help you practice using Delete, Move, Replace, and Insert with these nouns, we have provided a list of corrections to be made to the LETTER file and have suggested a way of making each correction. For the first five corrections, we show the details of what you type and what you see on your screen. After making the first five corrections, you should be familiar enough with the form of the commands to finish the rest of the corrections without seeing the details.

1. Remove the comma following "your demonstration" by using the Delete Character command and marking the comma. MiniBASE will leave the space between "demonstration" and "and."

You type	: Command window shows :
d	Delete C:
c	Delete Character (at) M/A:
<MARK>	Delete Character (at) ! OK:
<OK>	Delete Character (at) ! !
	BASE C:

2. Remove the word "indeed" between "certainly" and "presented." Use the Delete Word command and mark any character in "indeed." Notice how the text is adjusted when the word is deleted.

You type	: Command window shows :
d	Delete C:
w	Delete Word (at) M/A:
<MARK>	Delete Word (at) ! OK:
<OK>	Delete Word (at) ! !
	BASE C:

3. Change the word "presented" to "present" by deleting the "ed". Use the Delete Text command. Mark the "e" as the first character of the text and the "d" as the last character.

You type	: Command window shows :
d	Delete C:
t	Delete Text (at) M/A:
<MARK>	Delete Text (at) ! (through) M/A:
<MARK>	Delete Text (at) ! (through) ! OK:
<OK>	Delete Text (at) ! (through) ! !
	BASE C:

4. Change "Botoh" to "Booth" by moving the "t" to follow the second "o". Use the Move Character command. Mark the "t" as the character to be moved and mark the "o" as the character it should follow.

You type	:Command window shows :
m	Move C:
c	Move Character (from) M/A:
<MARK>	Move Character (from) ! (to follow character at) M/A:
<MARK>	Move Character (from) ! (to follow character at) ! OK:
<OK>	Move Character (from) ! (to follow character at) ! !
	BASE C:

5. Change the phrase "Our staff will later this week be meeting" to "Our staff will be meeting later this week". Use the Move Text command. For the text that you

want to move, mark the space before "later" as the first character of the text and the "k" in "week" as the last character of the text. For the character you want the text you are moving to follow, mark the "g" in "meeting".

You type	Command window shows
m	Move C:
t	Move Text (from) M/A:
<MARK>	Move Text (from) I (thru) M/A:
<MARK>	Move Text (from) I (thru) I (to follow) M/A:
<MARK>	Move Text (from) I (thru) I (to follow) I OK:
<OK>	Move Text (from) I (thru) I (to follow) I I BASE C:

6. Change the phrase "the involved details" to "the details involved". Use the Move Word command to move "involved" to follow "details" by marking any character in "involved" and then any character in "details".

7. Use the Replace Character command to replace the "Y" in "Yours" with "y" by marking the "Y" and then typing "y" followed by <OK> or marking any "y". Notice that if you mark a "y" to indicate what the character you want inserted, the "y" you mark will not be changed.

8. Change the phrase "planning to evaluate" to "evaluating". Use the Replace Text command. Mark the "p" in "planning" as the first character of the text you want to replace and then the "e" in "evaluate" as the last character of the text. Type the new text "evaluating" and then <OK>.

9. Use the Replace Word command to replace "present" with "attending" by marking any character in "present" and typing "attending" followed by <OK>.

10. Add a comma after "Sincerely yours" to make "Sincerely yours,". Use the Insert Character command and mark the "s" as the character the new character should follow; then type a comma followed by <OK> or mark a comma.

11. Add a new sentence "I hope this will be convenient." to follow the last sentence in the body of the letter. Use the Insert Text command and mark the period at the end of the last sentence. Then type "<SP><SP>I hope this will be convenient." and end with <OK>. You need to type the two spaces so there will be two spaces between the old sentence and the new one.

12. Change the phrase "a purchase" to "a possible purchase". Use the Insert Word command and mark the "a" as the word you want the new word to follow; then type the new word "possible" and end with <OK>.

If you see any other typing errors, use the editing commands you have just learned to correct them.

Remember that "Text" can be a single character. To replace one character with text showing on your screen, you can use the Replace Text command, mark the character as both the beginning and ending character of the text to be replaced, and then mark the text with which you want to replace it.

Marking words in MiniBASE has an additional feature. When you work with words, you can also mark the space between two words; MiniBASE will consider the two words as one. Or you can mark a hyphenated word by marking the hyphen between the words comprising it. Marking between words is often very useful in editing, and can significantly reduce the number of commands you have to give. For example, you could revise the phrase "I will be in touch with you Tuesday of next week" to read "I will be in touch with you next week", by giving the Delete Word command and marking the space between "Tuesday" and "of".

Why does this work? As an editing convenience, MiniBASE has been designed so that whenever you tell it you want to work with a word, it assumes that what you mark is a character in a word, without actually examining the character. This means that when you mark a space or hyphen between two words, MiniBASE assumes it is a character in the middle of a word and treats the two words as one. Similarly, you can move or insert a word to follow a punctuation mark immediately following a word, simply by marking the punctuation; MiniBASE assumes the punctuation is the last character in a word, places your new word after it, and inserts a space, as usual, before your new word.

There may be times when you want to move or insert a word to follow a punctuation mark that immediately precedes a word, such as a left parenthesis. When you mark the parenthesis, MiniBASE again thinks you have marked a character in a word (in this case, the first character) and inserts your new word after what it believes is the whole word. Thus your new word comes not only after the parenthesis but also after the word immediately following it.

NOTE: Line breaks do not affect what you can mark; spaces that fall at the ends of lines can be marked just as when they are within lines.

SAVING YOUR WORK: THE UPDATE COMMAND

All the work you've done so far is stored in the memory (RAM, or random access memory) of your personal computer. This kind of memory is called "volatile" memory, meaning that it can only retain information as long as the power is turned on. In other words, if at this point you turned your personal computer off (or if power was interrupted for any other reason), when you turned the computer back on your work would be gone.

You want to transfer your work to a more permanent form of storage -- either a floppy disk or perhaps (if your computer is equipped with one) a hard disk. You'll use the *Update File* command to do this. Once you use this command, your file will stay on the disk unchanged until you either modify it in a later work session, or delete it (using the DOS Delete command; see page 78).

The simplest form of this command is illustrated below:

You type	: Command window shows ;
u	Update C:
f	Update File T/OK:
<OK>	Update File ! OK:
<OK>	Update File ! !
	BASE C:

This simple form will work perfectly to save the work you've done on your sample business letter -- assuming that you began your work by using the Create File command as described above, and that you want to save this copy of your letter under the filename you gave your file when you used the Create File command. If you did NOT use the Create File command before you started your letter, but went ahead and used the simple form of the Update File command, you would have been rewarded with a message in the status window: "file won't open." The reason is that before you can save your work in a file you must give the file a name, using the following variation of the Update File command:

You type	: Command window shows ;
u	Update C:
f	Update File T/OK:
FILENAME	Update File FILENAME
<OK>	Update File FILENAME ! (in window) M/A:
<OK>	Update File FILENAME ! (in window) !
	BASE C:

The Update File command makes a new version of the file that incorporates all the changes you have made since you created or last updated it. If you enter a file name after "Update File," the new version will be saved under this file name. You should remember, though, that a properly formatted origin statement is created by the Create File command, not by the Update File command, so it's best always to begin your work on a new file with the Create File command.

File Names

File names can be up to eight characters long. This is a DOS restriction, not a MiniBASE one, but all software which runs under DOS must conform to it. In addition, most DOS files have three-character suffixes (e.g., FILENAME.ABC) in which the three-character extension helps DOS -- and you -- to keep track of what

kind of file it is. MiniBASE files are automatically given the extension "AUG"; other files have different extensions (e.g., .TXT, .COM).

It's generally a good idea to give your file a name that will help you to remember what's in it. For example, you might want to save the business letter you just wrote in a file called "LETTER." You could do this by typing "ufletter<OK><OK>" (for Update File LETTER); this would save your work in a file named LETTER.AUG.

Backup Files

Whenever you use the Update File command, MiniBASE actually does two things: it looks on the disk for the copy of the file which you loaded to begin work and renames it with the extension "AU@;" then it saves the new version of the file with an .AUG suffix. Thus, if you change your mind about some of the changes you made, you can return to the file as it was before you updated it by loading the .AU@ file (using the Jump File command, described below -- page 29) instead of the .AUG file.

Disk Drives

When you use the Update File command to save your work, MiniBASE decides which disk drive to store it on by these rules:

1. If you designate a disk drive (see below), the file will be stored on the designated drive.

2. If you do NOT designate a drive, and

   a. you give the file a name (e.g., you type "ufFILENAME<OK>"), the file will be stored on the CURRENT drive.

   b. you do NOT give the file a name because it is one you loaded from an earlier session, using the Jump (to) File command, the file will be saved on the disk drive from which it was loaded.

You can designate a different disk drive by entering the drive designator (the letter *a, b,* or *c* -- or occasionally *d*), followed by a colon and a filename, in the Jump (to) File command or the Update File command. For example, to save a copy of the file LETTER on the floppy disk in drive A, you would type "ufa:letter<OK><OK>"; the result would be "Update File a:letter ! !", with the desired effect.

More on Origin Statements

As we discussed above, the first statement in every MiniBASE file is called the origin statement. You will learn more about this statement later when you explore

file structure, but now is a good time to see what MiniBASE can do for you with this statement.

MiniBASE lets you do two different things with the origin statement: you can treat it like an ordinary statement, using it (for example) for the title of the file or document; or you can use it to store certain system-maintained information about the file and its history. The latter alternative is highly recommended if you plan to transfer files back and forth between MiniBASE and AUGMENT because it will help you to keep track of different versions of your files, and because it conforms to AUGMENT conventions.

When you update a file, MiniBASE checks the origin statement to see if it looks something like this:

<ABCD.AUG,>, 14-Oct-85 9:40 ;;;;

If it finds text in this format (a left angle-bracket, followed by some text (optional), followed by a right angle-bracket, followed by four semicolons), MiniBASE takes over maintenance of that statement as a "real" AUGMENT-style origin statement. The system writes and maintains the following information in the origin statement: <FILENAME.AUG,>, Date and Time (of last file update) ;;;;. This status information is updated every time you update a file.

Thus, if you entered as the first statement in your business letter file the text "<>;;;;" and then updated the file, naming it LETTER, the origin statement would look like this:

<LETTER.AUG,>, 16-Jul-85 10:38 ;;;;

The Create File command makes sure that your file does contain an origin statement in this format.

GETTING TO A FILE: THE JUMP (TO) FILE COMMAND

Suppose you leave MiniBASE after writing and editing a file as described in this lesson, and you later want to see the file again. Whenever you enter MiniBASE, you are presented with a blank work area; if you want to work in one of the files you created in an earlier session, you need to use a command to get to it. To reach any one of your files, use the Jump (to) File command. Simply type "jf" and then type the name of the file followed by <OK>. For example, this is how you could get to the file named LETTER after entering MiniBASE:

You type	Command window shows
j	Jump to C/M:
f	Jump (to) File T:
letter<OK>	Jump (to) File letter ! V:
<OK>	Your file LETTER.AUG appears on the screen.
	BASE C:

PRINTING YOUR WORK: THE PRINT COMMAND

MiniBASE was designed to provide basic hardcopy output capabilities, to allow you to produce drafts of large documents (up to MiniBASE's maximum capacity of 64K) and final copies of short documents (e.g., letters and memos).

If you need extensive printing or output formatting capabilities, you will need to use AUGMENT. Your MiniBASE User's Guide gives full information on moving files back and forth between your workstation and AUGMENT, while the Output Processor User's Guide provides complete information on AUGMENT's formatting capabilities; ask your McDonnell Douglas representative for help in learning more.

The Print command

The MiniBASE Print command is basically very easy to use, once you are set up. Each person has a personalized set of "configuration files" which tell MiniBASE exactly how he or she usually wishes to print. Thus every user could have a different type of printer, but all would use the same printing commands. These configuration files are maintained and modified using the *Setup* program, which is on your distribution disk. (A basic set of configuration files has been included on your MiniBASE diskette.) See the section on *Setup* in the *Reference* section below for more information.

For now, let us assume that all your configuration files are properly set up, and that you wish to print in your "usual" way.

All you need to do is type "p" for "Print" and then identify exactly what you wish to print -- a statement, branch, group, or plex. If you wish to specify viewspecs, you must type <OPT> (using the OPTION key) before typing "s" for "Statement", "b" for "Branch", and so on. Terminate the command with <OK> to begin printing.

Two simple cases of the Print command are presented below, together with a short explanation of any unusual features:

Print Branch (at) MARK <OK> <OK>

> The designated branch will be printed, using the current viewspecs and any margins or other printing parameters set up in the configuration file.

Print <OPT> (printing view)x<OK> Plex (at) MARK <OK> <OK>

> The "x" following the <OPT> is the viewspec with which the plex is to be printed. After the printing is complete, the viewspecs on your screen will remain as they were before the Print command was given (in other words, the "x" viewspec affects only the printed copy).

In the above cases, this is all that is required. Your document will be printed on your workstation printer. Of course, if your printer is unplugged or turned off, nothing will happen, but we were assuming that the hardware and configuration files were ready to go. A very large proportion of your printing will probably be taken care of by commands similar to those presented above.

For information on changing basic printing parameters, such as margins, see the section on configuration maintenance under "Setup" in the Reference section of this manual (page 81).

ENDING A MINIBASE SESSION: THE QUIT COMMAND

When you're finished work and are ready to leave MiniBASE, you should first make sure that you have updated the file in which you have been working, and then issue the Quit command by typing "q<OK>". You will be returned to the DOS command level, recognizable by the "A>" or DOS A:" prompt.

If you should attempt to Quit from MiniBASE without updating your file, MiniBASE will stop you with a message ("needs updating--aborted") warning you to update your file first. This prevents accidental loss of your work.

If you *really* want to quit anyway (if, for example, you've decided that the changes you made in that particular session aren't worth saving, or perhaps are even worse than the original), then you should jump to the top of the file and issue the *Delete Branch* command, marking the origin statement. You will then be able to Quit

successfully. (The Delete Branch command will be discussed below in the section on "Writing and Reading an Organized File," which begins on page 35.)

You type	:Command window shows :
j	Jump C/M:
o	Jump (to) Origin (of file) M/A:
<OK>	Jump (to) Origin (of file) ! V:
<OK>	(Origin statement of LETTER.AUG is at top of screen)
d	Delete C:
b	Delete Branch (at) M/A:
mark	Delete Branch (at) ! OK: (origin statement is marked)
<OK>	(the screen shows a blank work area)
q	Quit OK:
<OK>	DOS A:

SUMMARY

Several basic MiniBASE concepts were introduced in this lesson: using the verb-noun pattern to combine various command words; typing only enough characters in a command word for MiniBASE to recognize the word; understanding and responding to prompts; marking with the mouse or cursor (arrow) keys; and ending commands or parts of commands with <OK>. You have also learned how to cancel all or part of a command with <CD> or <BC>. These features are consistent throughout all of MiniBASE. Learning these elements will make it easy for you to learn the rest of MiniBASE.

This lesson has also taught you some basic commands for writing and editing in MiniBASE. You should now know how to create a file and add statements to it, how to use <BC> and <BW> to make corrections while you are typing text, and how to edit a statement in a file such as by removing or adding characters, words, or text. You have also learned how to update a file you have worked on and how to get to it again the next time you enter MiniBASE. We recommend that you take a break, and then proceed to the chapter on "Writing and Reading an Organized File".

LIST OF COMMANDS

Insert Statement (to follow) LOCATION LEVELADJUST TYPEIN<OK>

Insert Character (to follow) LOCATION CONTENT <OK>

Insert Text (to follow) LOCATION CONTENT <OK>

Insert Word (to follow) LOCATION CONTENT <OK>

Delete Character (at) LOCATION <OK>

Delete Text (at) LOCATION (through) LOCATION <OK>

Delete Word (at) LOCATION <OK>

Move Character (from) LOCATION (to follow) LOCATION <OK>

Move Text (from) LOCATION (through) LOCATION (to follow) LOCATION <OK>

Move Word (from) LOCATION (to follow) LOCATION <OK>

Replace Character (at) LOCATION (by) CONTENT <OK>

Replace Text (at) LOCATION (through) LOCATION (by) CONTENT <OK>

Replace Word (at) LOCATION (by) CONTENT <OK>

Update File <OK> <OK>

Update File <OK> FILENAME <OK>

DEFINITIONS

LOCATION -- Prompted by "M/A:"

For M you may <MARK>.

CONTENT -- Prompted by "M/T:"

For M you may <MARK>.

For T you may type a series of characters, ending with <OK> (if another <OK> follows, a second one is not needed).

TYPEIN -- Type a series of characters, ending with <OK>.

WRITING AND READING AN ORGANIZED FILE

INTRODUCTION

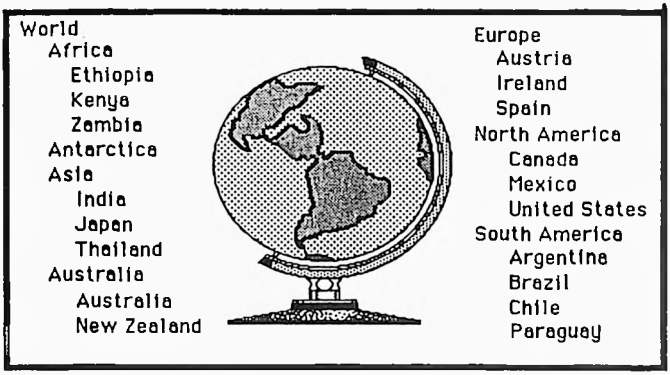
Writing in MiniBASE means adding text to a file, either by typing at your keyboard or by copying text already stored online. When you add to a MiniBASE file, you can organize the paragraphs and headings into an outline form that reflects the relationships between the statements. For example, you can enter a document in which the first chapter title will be at position 1, the first subheading in that chapter at position 1a, the first paragraph under that subheading at position 1a1, and so on. This outline form is called "hierarchical structure". This lesson will teach you about hierarchical structure and tell you how to write a file that is organized in this way. You will see how hierarchical structure helps you organize both your thoughts and your writing.

You will also learn how hierarchical structure makes it possible for you to look at an online document from many different points of view. Reading in MiniBASE is much more flexible than reading text printed in a book. When reading a book you are limited by the fact that the content of each page is fixed; you have only two choices, to read sections straight through, from beginning to end, or to scan, picking out sentences here and there. In MiniBASE you can read the paragraphs on your screen, one after the other, as if they were on the pages of a book, but you can also look at a document in many other useful ways. Since you control how you see a file on the screen, you can take advantage of the structure of MiniBASE files by looking at only headings, for example, or only headings and the first line of every paragraph; the other text is still in the file, but you look at only what you need to see.

ORGANIZED FILES: HIERARCHICAL STRUCTURE

People tend to classify the things they see around them. One way of classifying things is in hierarchies. (Another way would be in groups of similar objects, for example, all chairs, or all land masses.) Classifying in hierarchies is an important kind of classification because hierarchies allow you not only to talk about groups of things but also to show the relationships between things. For example, it is common to think of the world itself in terms of a hierarchy, as follows: The world is divided into continents, the continents are divided into countries, countries are divided into states, states into counties, and so on.

Because hierarchical arrangement is such an important tool, MiniBASE files are structured so that you can organize statements into an outline form that shows the relationships between the statements. Here is an example showing how you would express the hierarchical organization of the world with MiniBASE structure.



Each item in this outline is a separate statement. Note that although each statement shown here consists of one or two words, a statement in a hierarchy can have any content: a character, a number, a word, a line, a sentence, or a paragraph. What is important in setting up the structure is the relationships between the statements. MiniBASE shows these relationships with indenting. Each statement that is indented from the preceding statement is subordinate to that statement.

The hierarchy in the above outline has three "levels": the World statement is at level 1, the continents are one level below it, at level 2, and the countries are at level 3. We could fill in this outline by including more information at level 3 (for example, adding the rest of the countries under each continent), or we could add more levels (by adding states under countries, counties under states, and so on, until we reached streets or even individual buildings).

To describe the relationships between statements at different levels, we use the words "up" and "down." Statements at level 1 (also called "first-level statements") are one level "up" from statements at level 2 ("second-level statements"). Second-level statements are one level "down" from first-level statements and one level up from third-level statements. Likewise, third-level statements are one level down from second-level statements, two levels down from first-level statements, one level up from fourth-level statements, and so on.

Two more MiniBASE terms used to describe the relationships between statements are "substatement" and "upstatement." A statement's "substatements" are the statements that are subordinate to it and one level down from it. In the preceding example, Africa has the substatements Ethiopia, Kenya, and Zambia, and Antarctica has no substatements. "Upstatement" is the reverse of substatement; a statement's upstatement is the statement it is subordinate to and one level down from. Thus the upstatement for Ethiopia, Kenya, or Zambia is Africa, and Antarctica's upstatement is World.

We also refer to the "substructure" of any MiniBASE statement that has substatements. A statement's substructure is all the statements subordinate to it, regardless of how many levels they are down from it. In other words, the substructure of a statement is all of its substatements, plus all of their substatements, plus their substatements, and so on. For example, the substructure of the World statement is the entire rest of the outline, including all the continents and the countries under each continent. The substructure of the continent Australia consists of the countries Australia and New Zealand. Not every statement has substructure. Since Antarctica has no countries under it, it has no substructure, and since none of the individual countries have statements under them, none of them has substructure.

This lesson will show you how to write a structured MiniBASE file containing a table of contents, a familiar example of hierarchical structure. When you know how to insert this table of contents, you will know everything necessary to write any structured file you wish.

NOTE: We have chosen an example that consists of only headings, for brevity; of course, the files you write will normally include information under the headings.

To create a file named BENEFITS that is to contain this table of contents, you would enter MiniBASE and insert an origin statement at the top of your blank work area. Use the Create File command to do this (see the section on "Origin Statements" in the first lesson, above), naming your file "benefits". When you do this, MiniBASE automatically makes and displays an origin statement containing information about the file. You would then be ready to insert the table of contents.

The BENEFITS Table of Contents

```
Contents
   Part I.  Holidays
      Section I.  Legal days off.  List of legal
      holidays, including national, local, and
      company-specific days.
      Section II.  Absence from work.  Rules
      governing permissible days off with and
      without pay.
   Part II.  Recreational Activities
      Section I.  Noon time.  Location for classes
      and dressing rooms and regulations concerning
      their use.
      Section II.  Scheduled activities.  List of
      available classes and location for each
      calendar year.
      Section III.  Company-sponsored classes.  Off-
      site classes and registration information.
   Part III.  Education
      Section I.  Qualification.  Information
      regarding hours and requirements for work-
      related classes.
      Section II.  Costs.  Regulations regarding
      reimbursement for registration and tuition
      fees.
      Section III.  Advanced degrees.  Applications,
      requirements, and program development.
   Notes
   Addendum
```

INSERTING STATEMENTS AT DIFFERENT LEVELS

To insert statements at different levels, as you would need to do to enter the sample table of contents, you can use the *Insert Statement* command. After creating the BENEFITS file that is to contain the table of contents, you could begin by inserting the first statement to follow the origin statement of the file. You would do this by giving the Insert Statement command and marking the origin statement as the statement your new statement should follow.

You type	Command window shows
i	BASE Insert C:
s	BASE Insert Statement (to follow) M/A:
<MARK>	BASE Insert Statement (to follow) I L:
<OK>	BASE Insert Statement (to follow) I M/T:
Contents<OK>	BASE Insert Statement (to follow) I Contents!
	BASE C:

As shown in this example, when you insert a statement you see the prompt "L:" and can respond by <OK> and then typing in your new statement. However, you have another choice. Before entering the new statement, you can indicate that you want to put it on a different level than the previous one. The "L:" means that you can specify the level of the new statement relative to the level of the statement it is to follow; we call this "adjusting the level" of the new statement. This ability to put statements at different levels as you enter them allows you to arrange the statements in your file into a hierarchy as you type.

In response to the "L" prompt you can specify that your new statement should be down from, up from, or at the same level as the statement it is to follow. To indicate that your new statement should be down a level (and therefore a substatement of the statement it is to follow), type "d<OK>". To put your new statement up a level, type "u<OK>"; it will be inserted at the next highest level (that is, the same level as the upstatement of the statement it is to follow). If you want the statement to be up more than one level, type another "u" for each additional level. For example, "uuu<OK>" will put your new statement three levels higher than the one you insert it to follow. If you want your new statement to be at the same level, you can type <OK> in response to the "L", or you can simply begin typing; if you do not specify a different level for your new statement, MiniBASE will insert it at the same level as the one it follows.

NOTE: You cannot insert a statement to be down more than one level from the one it is to follow.

There is one place where you do not need to indicate a level change at the "L:" prompt, even though the statement you insert will be at a different level than the one it follows. You do not need to specify a level change when you are inserting to follow the origin statement; you can either begin typing, or you can type an <OK> and then begin typing. The origin statement is the upstatement of all the first-level statements you add to a file; in other words, the origin statement is at level 0 and all the other statements in the file are the substructure of the origin statement. MiniBASE knows this and thus automatically puts at level 1 any statement you insert to follow the origin statement. When inserting the first statement as shown in the previous example, you took advantage of this when you responded to the "L:" prompt by typing <OK> followed by your new statement. Although you did not tell MiniBASE to change the level of the new statement, MiniBASE made it a first-level statement, one level lower than the origin statement.

Inserting to follow the origin statement is the only place MiniBASE will automatically adjust the level of your new statement. When entering the second statement of the table of contents, for example, you will have to adjust the level yourself. To do this, give the Insert Statement command again and mark the first statement you inserted, "Contents", as the statement your new statement is to

follow. At the "L:" prompt, type "d<OK>" to indicate that your new statement should be down a level from "Contents". You will see a new prompt, "M/T:". MiniBASE is waiting for your new statement; type "Part I. Holidays" and end with <OK>. The process of entering the second statement looks like this:

You type	: Command window shows ;
i	BASE Insert C:
s	BASE Insert Statement (to follow) M/A:
<MARK>	BASE Insert Statement (to follow) I L:
d<OK>	BASE Insert Statement (to follow) I dI M/T:
Part I.	BASE Insert Statement (to follow) I dI
Holidays<OK>	Part I. HolidaysI
	BASE C:

After the final <OK>, MiniBASE will display the statement at the place and level you have indicated. The new statement will follow the previous one immediately, with no blank line between them. (Later in this lesson you will learn how to see the file with blank lines between the statements.)

You are now ready to enter Sections I and II of Part I. Because these are logically subsections of Part I, you would want to take advantage of MiniBASE's hierarchical structure and make them substatements of Part I. You could mark Part I as the statement you want your new statement to follow and insert Section I down a level from it, as follows:

You type	Command window shows ;
i	BASE Insert C:
s	BASE Insert Statement (to follow) M/A:
<MARK>	BASE Insert Statement (to follow) I L:
d<OK>	BASE Insert Statement (to follow) I dI M/T:
I. Legal....	BASE Insert Statement (to follow) I dI I. Legal days off.
.....	List of legal holidays, including national, local, and
...days.<OK>	company-specific days.I
	BASE C:

You could then mark Section I as the statement your next new statement should follow and insert Section II. Since Section II is to be at the same level as Section I, you do not have to adjust the level. Thus at the "L:" prompt you could just begin typing your new statement, ending it with <OK>.

You type	Command window shows
i	BASE Insert C:
s	BASE Insert Statement (to follow) M/A:
<MARK>	BASE Insert Statement (to follow) I L:
<OK>	BASE Insert Statement (to follow) I I M/T:
Section II.	BASE Insert Statement (to follow) I Section II. Absence
... from work.	Rules governing permissible days off with
....pay<OK>	and without pay.I
	BASE C:

If you find it more convenient or easier to type, you can use <SP> instead of <OK> when you adjust the level of your new statement. For example, "u<SP>" would put your new statement up one level and <SP> would put it at the same level.

Once you have finished entering the Sections under Part I, you can begin entering Part II. Since the Part II statement is logically on the same level as the Part I statement, one level higher than the Sections under Part I, you can insert it to follow Section II of Part I up a level.

You type	Command window shows
i	BASE Insert C:
s	BASE Insert Statement (to follow) M/A:
<MARK>	BASE Insert Statement (to follow) I L:
u<OK>	BASE Insert Statement (to follow) I ul M/T:
Part II.	BASE Insert Statement (to follow) I ul Part II.
...<OK>	Recreational Activities!
	BASE C:

At this point you could continue giving Insert Statement commands, marking the statement your new statement should follow, adjusting the level as necessary, typing the statement, and ending with <OK>, until you have entered the entire table of contents. But a faster way to do this would be to use "insert mode," as described below.

INSERTING A SERIES OF STATEMENTS USING INSERT MODE

Whenever you want to insert a series of statements where each new statement directly follows the previous one, you can, of course, give a separate Insert Statement command for each statement; however, it saves time to use "insert mode." In insert mode you do not have to give the Insert Statement command over and over; instead, you just type your statements one after another, adjusting the level when necessary, and the statements are inserted into your file in the order you type them.

To get into insert mode, you type <INS>. (See the Keyboard Equivalency Table on page 3 to find the Insert -- <INS> -- key for your workstation device.) The

following paragraphs tell where you can do this. To leave insert mode, you type <CD>, as you would to cancel a command.

You can type <INS> at "BASE C:" after you have given an Insert Statement command; MiniBASE will assume that you want to begin inserting right after the statement you just entered. When you type <INS>, you see an "L:" prompt. This is the same prompt as in the Insert Statement command, and you should respond to it in the same way: Adjust the level of your new statement as necessary, type the statement, and end with <OK>. MiniBASE will insert the statement into your file and prompt you again with "L:". You can then enter your next statement to follow the one just inserted, adjusting the level as necessary, and following this with the next statement, and so on, until you have entered all the statements. When you are finished, type <CD> at the "L:" prompt.

Continuing with the example of inserting the BENEFITS table of contents, suppose you have just entered the Part II statement with the Insert Statement command and you want to use insert mode to enter the remaining statements.

You type	Command window shows
<INS>	L:
d<OK>	dI M/T:
Section I.	dI Section I. Noon time. Location for classes and
...<OK>	dressing rooms and regulations concerning their use.I
	L:
<OK>	I M/T:
Section II.	Section II. Scheduled activities. List of available
...<OK>	classes and location for each calendar year.I
	L:
<OK>	I M/T:
Section III.	Section III. Company-sponsored classes. Off-site
...<OK>	classes and registration information.I
	L:
u<OK>	uI M/T:
Part III.	uI Part III. EducationI
...<OK>	L:
<CD>	BASE C:

Notice that just as when you use the Insert Statement command, you should respond to the "L:" prompt either by adjusting the level of your new statement (as when entering the Section I and Part III statements above) or, if the new statement is at the same level as the one it follows, by typing <OK> followed by the text of the statement (as when entering Sections II and III), or by simply beginning to type.

You could continue in this way to enter the entire table of contents; however, you would not see all of it in your file window. At some point the file window would be too small to display all the statements in your file. Later in this lesson you will learn

how to see the rest of your file; right now it is important to realize that as long as you type each statement and end it with <OK>, it will be added to your file whether or not you can see it being added. Of course, the command window will always show the statement you are currently typing.

When entering the "Notes" statement, be sure to type "uu<OK>" after the "L:" prompt, because this statement is two levels up from the statement it follows (Section III of Part III). Remember that to leave insert mode you must type <CD> after the "L:" prompt.

While it is perhaps safest to enter insert mode after giving an Insert Statement command, you can, in fact, type <INS> any time you are prompted with "BASE C:". When you type <INS>, MiniBASE will assume you want to begin inserting after your current statement, that is, the last statement you inserted, edited, or jumped to. If you are inserting into a newly created file, or just beginning to work, the origin statement will be your current statement and the first statement you insert will follow the origin statement. Because it may be difficult at times to determine just what your current statement is, it is a good practice to begin inserting with the Insert Statement command. After the Insert Statement command, when you know where you are, you can safely enter insert mode.

You can also type <INS> in place of the final <OK> in an Insert Statement command. Because you are replacing <OK> with <INS>, you will not see the exclamation point (!) that lets you know MiniBASE has received an <OK>. MiniBASE will carry out the Insert Statement command as usual and then give you an "L:" prompt just as if you had typed <INS> at "BASE C:". Type <OK> at the end of each statement you insert, and type <CD> after the "L:" prompt when you are done.

INSERTING INDIVIDUAL STATEMENTS ANYWHERE

After entering a series of statements into a file, you may find that you want to add statements to what you have entered. To add statements anywhere in a file, use the Insert Statement command, mark the statement you want the new statement or statements to follow, and proceed as usual. If you are adding one statement, end the Insert Statement command with <OK>; if you are adding a series of statements, you may want to enter insert mode by typing <INS> either in place of the final <OK> or at "BASE C:" after the Insert Statement command. For example, to add an "Introduction" before Part I in the BENEFITS table of contents, you could use the Insert Statement command, mark the Contents statement, and add the Introduction statement to follow it, down one level.

When you insert a statement to follow another at the same level and the statement it is to follow has substructure, MiniBASE will put the new statement after the sub-

structure. If, for example, you added a Part IV statement to the table of contents and indicated that it should follow Part III at the same level, MiniBASE would put your new Part IV at the same level as Part III, but after Section III of Part III. (You will be able to check this later in this lesson, after you learn about reading and moving around in a file.)

Note that there is often more than one way to add a statement at a particular position in a hierarchically structured file. For example, if you inserted a Part IV statement to follow Section III of Part III and indicated that it should be up one level, the result would be the same as if you inserted it to follow Part III at the same level. In fact, the same thing would happen if you inserted a statement to follow Section I or II of Part III up one level.

In general, we can say that when you insert a statement MiniBASE puts your new statement in the next available place at the level you specify, without disturbing the substructure of the preceding statement at that level. Thus, inserting a statement to follow a third-level statement up one level puts the new statement in the next available place at level 2, inserting it up two levels puts it in the next available place at level 1, and inserting it up three levels puts it immediately after the origin statement.

READING: THE JUMP COMMAND

If you have added statements to a file at a place that is not displayed on your screen, you will probably want to look at what you have added. To read statements that are not displayed on your screen, you can use one of the many forms of the *Jump* command. This command is called "Jump" because it lets you jump from one statement to another without reading through all the statements between them. When you jump to a certain statement, MiniBASE displays that statement at the top of the file window and shows as much of what follows as will fit on the screen. MiniBASE offers many ways of jumping around in a file and helping you find a particular statement. In this lesson you will learn several basic kinds of Jump commands; later you will learn others.

After you type "j" for "Jump," your command window will show "Jump (to)" followed by the prompt "C/M:". MiniBASE is asking you to specify where you want to jump. You can respond to the "M" in this prompt by *marking* the statement to which you want to jump, or to the "C" in this prompt by giving a command word describing where you want to jump.

The simplest way to indicate the statement to which you want to jump is to mark it. After marking the statement, you will see a "V:" prompt. This prompt means that you may change viewspecs (as described later in this lesson). To keep the same type of view, simply type <OK>. The statement you marked will then move to the top of

the file window and you will see the series of statements that follow it. For example, after inserting the table of contents into the BENEFITS file, you could read beyond what shows on your screen by using Jump and marking any character in the last statement in your file window.

You type	Command window shows:
j	BASE Jump (to) C/M:
<MARK>	BASE Jump (to) ! V:
<OK>	BASE Jump (to) ! !
	BASE C:

Notice that no matter what character you mark in a statement, the Jump command always moves the beginning of that statement to the top of the file window.

After reading the entire table of contents, you could return to the beginning of the file by using the *Jump (to) Origin* command. When you see the "C/M:" prompt after typing "j" for "Jump," respond to the "C" by typing "o" for the command word "Origin," indicating the origin statement of the file. Then mark any character in the file and end with an <OK> after the "V:" prompt. MiniBASE will display the origin statement of your file at the top of the file window.

You type	Command window shows
j	BASE Jump (to) C/M:
o	BASE Jump (to) Origin (of file) M/A:
<MARK>	BASE Jump (to) Origin (of file) ! V:
<OK>	BASE Jump (to) Origin (of file) ! !
	BASE C:

JUMPING WITH STRUCTURE: NEXT, BACK, SUCCESSOR, AND PREDECESSOR

Four important command words you can use with the verb "Jump" to get to a statement not displayed in the file window are *Next*, *Back*, *Successor*, and *Predecessor*. Like "substatement" and "upstatement," these terms describe structural relationships between statements in a MiniBASE file. When you use these command words, you must indicate a statement to serve as a reference point so that MiniBASE will know which statement the one you want is next from, back from, and so on.

The *next* statement from the statement you indicate is the statement immediately following it, and the statement *back* from the statement you indicate is the statement immediately preceding it, regardless of level. For example, to jump to the statement following the last statement in your file window, you can give the *Jump (to) Next* command and mark this last statement. In order to make MiniBASE command structures consistent with those of AUGMENT, you must type "<SP>n" for "Next."

You type	:Command window shows
j	BASE Jump (to) C/M:
<SP>n	BASE Jump (to) Next (from) M/A:
<MARK>	BASE Jump (to) Next (from) I V:
<OK>	BASE Jump (to) Next (from) I I
	BASE C:

If you then give the *Jump (to) Back* command and mark the statement at the top of your file window, the immediately preceding statement will appear at the top of the window.

You type	:Command window shows
j	BASE Jump (to) C/M:
b	BASE Jump (to) Back (from) M/A:
<MARK>	BASE Jump (to) Back (from) I V:
<OK>	BASE Jump (to) Back (from) I I
	BASE C:

Every statement except the last statement in a file has a "next" statement, and every statement except the origin statement has a statement that is "back" from it.

The *successor* of a statement is the next statement that is at the same level and has the same upstatement. Not every statement has a successor. If a statement is the last or only substatement under a particular upstatement, it has no successor. The *predecessor* of a statement is the preceding statement that is at the same level and has the same upstatement. Just as not every statement has a successor, not every statement has a predecessor. If a statement is the first or the only substatement under a particular upstatement, it has no predecessor.

Jumping to the successors and predecessors of statements will let you quickly view the various sections of a document that are at the same level. If, for example, the Notes statement of the BENEFITS table of contents were showing on your screen, and you gave the *Jump (to) Predecessor* command and marked that statement, you would see "Contents" at the top of the file window.

You type	Command window shows
j	BASE Jump (to) C/M:
p	BASE Jump (to) Predecessor (of) M/A:
<MARK>	BASE Jump (to) Predecessor (of) I V:
<OK>	BASE Jump (to) Predecessor (of) I I
	BASE C:

If you then jumped to the successor of "Contents," you would see "Notes" at the top of the file window.

If you try to jump to Next, Back, Successor, or Predecessor when there is none relative to the statement you indicate, the statement you indicate will appear at the

top of the file window. For example, since "Addendum" is the last first-level statement in the BENEFITS file, you could not jump to its successor; if you tried to, MiniBASE would display "Addendum" itself at the top of the file window.

JUMPING BACK IN TIME: THE JUMP RETURN COMMAND

The Jump commands we've discussed so far help you to move around through different locations in the file. Sometimes, though, you don't remember the exact location you want; instead, you want to go back to a previous location. Perhaps it's the one you just left (and you suddenly remember something you forgot to do there), or perhaps it's farther back.

Fortunately, MiniBASE remembers the last ten locations you've visited in your current file, and allows you to return to them with the *Jump Return* command.

You type	Command window shows
j	BASE Jump (to) C/M:
r	BASE Jump (to) Return OK:
<OK>	BASE Jump (to) Return ! (previous location) Y/N:
y	(previous location) ! BASE C:

Actually, it won't say "previous location;" instead, you'll see the first few words of the last location. Responding with "y" to the "Y/N:" prompt will return you to that location. If you respond with "n" for No, you'll be offered another location (one location back in time from the one you just declined), and can say yes or no to that one -- and so on, back through a "return ring" of 10 locations.

CHANGING VIEWS: VIEWSPECS

MiniBASE enables you to control not only what part of your file you see but also how you see it. Just as you can jump around in a file using the various Jump commands to look at only the information you need, you can also control the way information is displayed on your screen. You can, for example, display your file with blank lines between the statements. When you view your file with blank lines, it does not mean MiniBASE has put spaces into your file; MiniBASE merely displays your file in a different way, leaving its contents just as you entered them.

The way you tell MiniBASE how you want to see the information in your file window is by setting *viewspecs*. A "viewspec" is a single letter that specifies the kind of view you want of your file. Some of the viewspecs that are currently controlling your view are displayed in the viewspec window (the upper right corner of your screen). When you enter MiniBASE, certain viewspecs are automatically in effect, such as those that let you see all lines of all statements in

your file. You can change your view any time you use the Jump command by responding to the "V:" prompt with the appropriate viewspec character.

You can also use the *Set Viewspecs* command to change your view of the file. Type "<SP>sev", followed by the viewspecs you want (see below).

Viewspecs are an important tool in working with your files. By using viewspecs effectively you can make working and reading, as well as searching your files, much easier. The rest of this lesson will introduce you to some basic viewspecs.

VIEWING BLANK LINES BETWEEN STATEMENTS

Viewspecs *y* and *z* allow you to control whether MiniBASE displays blank lines between statements.

Viewspec:	Means
y	Blank lines between statements on
z	Blank lines between statements off

Many people find reading files much easier with viewspec *y*. You can change your view in other ways, by setting other viewspecs, while keeping the blank lines between statements on. To see the file again without blank lines, just use viewspec *z* instead of viewspec y in your next Jump command.

LEVEL AND LINE CLIPPING VIEWSPECS

To get an overall picture of a book, a reader often looks at a list of the chapter titles in the table of contents. In MiniBASE, you can easily display only the headings in a document by using viewspecs to show only the statements at level 1. You can also show only the statements at levels 1 and 2, or only those at levels 1 through 3, and so on. This is called "level clipping." In a similar way you can control the number of lines displayed for each statement. This is called "line clipping." Because line clipping viewspecs let you fit more statements on the screen, they are useful in scanning the overall structure of a file. You will find level and line clipping viewspecs very helpful in both writing and reading.

As you learn to use viewspecs to change the way you see your file, it will help you to look at the viewspec window, which tells you how many levels and lines are being displayed. The number on the left indicates the number of levels displayed and the number on the right indicates the number of lines displayed. When you enter MiniBASE, the viewspec window shows "ALL ALL", meaning that you will see statements at all levels and all the lines of every statement.

Level Clipping Viewspecs

You can use level clipping viewspecs to display statements at the first level of your file, the first two levels, the first three levels, or any number of levels up to 63. These are the level clipping viewspecs:

Viewspec:	Means
a	Show one level less
b	Show one level more
c	Show all levels
d	Show first level only

For a view of your file that shows only first-level statements, you would use viewspec *d* to "show first level only." If you use viewspec *d* and have not set any line clipping viewspecs, you will see "1 ALL" in the viewspec window, meaning that one level and all lines are being displayed.

After setting viewspec *d*, you can use viewspec *b*, which means "show one level more;" you will then see statements at levels 1 and 2 in the file window and "2 ALL" in the viewspec window. If you looked at the BENEFITS file after setting viewspecs *d* and *b*, you would see the statements at levels 1 and 2, and you would not see the "Section" statements (which are at level 3). To display three levels of statements, you could add another viewspec *b* to "show one level more." Similarly, to display four levels, you would add still another viewspec *b*, and so on. Each *b* will add a level to what you had before.

Viewspec *a* means "show one level less." If statements at three levels were being displayed and you wanted to see only the statements at levels 1 and 2, you could use viewspec *a*.

It is important to understand that level clipping viewspecs display a number of levels more or less than the number in effect from previous viewspecs, not necessarily a number of levels more or less than what you see on your screen. For example, ALL levels is 63 levels. Thus, if you set viewspec a when you have ALL levels showing, you will have 62 levels; this will change your view only if the part of the file you are looking at contains a statement at level 63.

NOTE: Whenever you clip levels and the statement at the top of the file window is at a lower level than what you want to see, MiniBASE will continue to show that statement and any statements immediately following it at the same level; depending on the structure of your file, other statements at clipped levels may also appear before the first statement that is at the level to which you limited your view. For example, if you use viewspec *d* to "show first level only" when there are two second-level statements at the top of your file window, you will see these second-level statements and then only first-level statements after them.

To return to a view showing statements at all levels in your file, you can use viewspec *c*. This viewspec is in effect when you enter MiniBASE.

Line Clipping Viewspecs

You can use line clipping viewspecs to see the first line of the statements displayed, the first and second lines, the first three lines, and so on, up to all lines of each statement displayed. These are the line clipping viewspecs:

Viewspec:	Means
q	Show one line less
r	Show one line more
s	Show all lines
t	Show first lines only

As you can see from this table, line clipping viewspecs work much like level clipping viewspecs, although of course they control different things. Level clipping viewspecs control which statements you see, but do not affect the number of lines you see of each statement. With line clipping viewspecs, on the other hand, you can control how many lines you see of each statement, but you cannot control which statements are displayed. The actual statements displayed will be those at the levels you have selected with your level clipping viewspecs.

To see only the first line of each statement displayed, use viewspec *t*. If you use viewspec *t* when statements at all levels are being displayed, your viewspec window will show "ALL 1", meaning "all levels and one line." This view will allow you to fit more statements on the screen and is useful when you want an overview of the organization of a file or need to see the exact position and context of individual statements or parts of a document.

After setting viewspec *t*, you can use viewspec *r* to "show one line more;" you will then see the first two lines of every statement displayed. Or you could see three lines by using *r* twice after setting viewspec *t*, and so on, adding an *r* for each additional line you wanted to see.

To "show one line less," use viewspec *q*. For example, if you have three lines showing and you set viewspec *q*, the last line will disappear and you will see only the first two lines of each statement displayed.

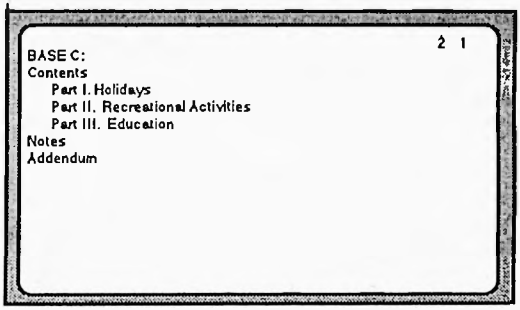
Like level clipping viewspecs that show one level more or one level less, line clipping viewspecs that show one line more or one line less add to or subtract from the number in effect from previous viewspecs, not necessarily from what you see on your screen. The maximum number (shown as "ALL" in the viewspec window) is 63.

When you want to return to a full view of every statement displayed, you can use viewspec *s* to "show all lines." This viewspec is in effect when you enter MiniBASE.

Combining Level and Line Clipping Viewspecs

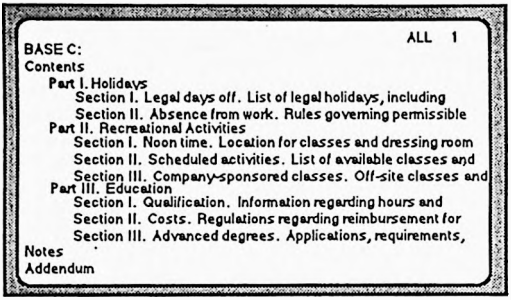
Level and line clipping viewspecs are very handy when used separately, but they can be even more effective when used together. Suppose, for example, you are working with a large file, and you decide to use viewspec *d* so you can scan the statements at level 1. If you show every line, you may not be able to see all the first-level statements at once, because they may not fit on the screen. To see as many of the statements as possible, you could add a line clipping viewspec; in addition to viewspec *d*, you could use viewspec *t*. Your viewspec window would then show "1 1", meaning "one level and one line." You would see only the first lines of the first-level statements in your file, so more of your file would fit on the screen.

If, after looking at your file with viewspecs *d* and *t*, you wanted to add another level to your view, you would use viewspec *b*. You would then see the first lines of the statements at levels 1 and 2 in your file and your viewspec window would show "2 1", for "two levels and one line." This type of view is useful when you are writing or reading a document and you want to see only the major headings (at level 1) and the subheadings under them (at level 2). For example, this is how the original BENEFITS table of contents would look with this view:



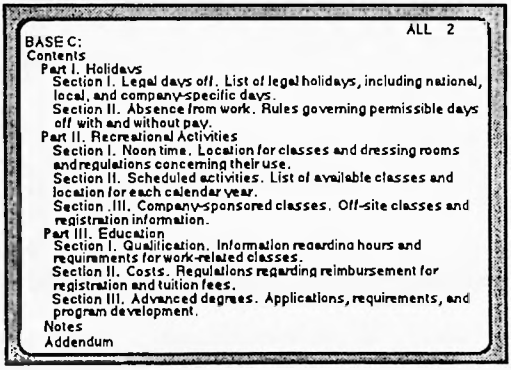
Once you have seen your file with viewspecs *d*, *b*, and *t*, you might want to display the first lines of all the statements in the file so that you can scan its structure. You would do this by using viewspec *c* to "show all levels;" MiniBASE would continue to show only one line of each statement displayed, and you would see "ALL 1" in the viewspec window.

Continuing from the previous example, this is what the table of contents would look like if you then used viewspec *c*:



NOTE: The line length shown here may differ from what you see on your terminal.

If you then wanted to add a second line to the statements being displayed, you would use viewspec *r*. MiniBASE would show one more line of each statement, and you would see "ALL 2" in the viewspec window. An additional viewspec *r* would change the above view to this:



Since no statement consists of more than two lines, this view shows you the full table of contents. (As when you first entered the statements into the file, you might not

be able to see all of them at once on your screen.) Note that if you then inserted a statement of more than two lines into this file, only the first two lines of it would appear in the file window.

Once you had this view, you could take away the third-level statements using viewspec *a*, or subtract a line using viewspec *q*, and so on, continuing to combine the level and line clipping viewspecs as desired. Because the combination of level and line clipping is so helpful and so frequently used, MiniBASE offers the following pair of viewspecs:

Viewspec	Means
w	Show all levels and all lines
x	Show level one and one line only

Viewspec *x* combines viewspecs *d* and *t*. Using viewspec *x* will display the first lines of the first-level statements in your file and will cause your viewspec window to show "1 1". Viewspec *x* not only combines level and line clipping, but can itself be conveniently used in combination with other level and line clipping viewspecs. For example, to show the first lines of the statements at levels 1 and 2 in your file, you could set viewspecs *x* and *b*.

This will have the same effect as setting viewspecs *d*, *b*, and *t*. Likewise, if you want to see the first two lines of the statements at level 1, you can use viewspecs *x* and *r*, which would have the same effect as setting viewspecs *d*, *t*, and *r*.

Just as viewspec *x* is a combination of viewspecs *d* and *t*, viewspec *w* is a combination of viewspec *c*, which shows all levels, and viewspec *s*, which shows all lines. Setting viewspec *w* is an easy way of counteracting any level and line clipping you may have done and restoring the "all levels and all lines" view you had when you entered MiniBASE. When you use viewspec *w*, you will again see "ALL ALL" in your viewspec window.

SEARCHING BY JUMPING AND CHANGING VIEWS

Changing viewspecs while you jump is a good way to look for something in a structured file. You could first show only one line of the statements at level 1, then jump to the heading that interests you and give a viewspec that adds another level, check the subheadings and jump to one with a viewspec that adds another level, and so on, until you find the statement you want to read. You could then jump to that statement with a viewspec that shows all lines and all levels so that you could read the entire paragraph and whatever follows it.

Suppose, for example, that you were unfamiliar with the BENEFITS file and needed to locate the section dealing with advanced degrees. You could begin by viewing the file with viewspec *x*. You would then see that the three main parts of

the file are Contents, Notes, and Addendum, and you would want to look further under Contents. You could then jump to "Contents" and at the same time set viewspec *b* to see another level of statements, that is, the "Part" statements. After noting that Part III deals with education, you could then jump to that statement, setting viewspec *b* again to add yet another level. The "Section" statements would be displayed under Part III and you would see that Section III of Part III was the section you wanted. Finally you could jump there, setting viewspec *w* to show all lines and all levels; MiniBASE would display the entire statement, and you would be able to read or modify it as you wished.

REVIEW OF VIEWSPECS

You have now learned many of the most frequently used and convenient of the viewspecs MiniBASE offers. The following table is intended as a review of the viewspecs presented in this lesson and as a quick reference table. For a complete explanation of the individual viewspecs, see the sections introducing them.

Viewspec	Means
y	Blank lines between statements on
z	Blank lines between statements off
a	Show one level less
b	Show one level more
c	Show all levels
d	Show first level only
q	Show one line less
r	Show one line more
s	Show all lines
t	Show first lines only
w	Show all levels and all lines
x	Show one level and one line only

Whenever you set viewspecs, they either modify or counteract the viewspecs already in effect. For example, viewspec *b* modifies the view you get with viewspec *d* (by adding a level); and you can use viewspecs *b* and *r* to modify the effect of viewspec *x* (by adding a level and a line).

If you use a single Jump or Set Viewspecs command to set several viewspecs, the result will be the same as if you had set each viewspec with a separate command. If the viewspecs you set modify each other, MiniBASE will interpret the viewspecs in the order that you type them, modifying each viewspec by what follows it. For example, if you type "xbr" and then, before ending with <OK>, you realize that you do not actually want to set viewspec *b*, you can type "a" at that point; viewspec *a* will take away the additional level added by viewspec *b*, and you will see two lines of only statements at level 1. Likewise, if you specify two viewspecs that contradict

each other, such as viewspec *d* to show the first level only and viewspec *c* to show all levels, the last viewspec you type will take effect. So if you set viewspecs "dc", viewspec *c* will take effect and viewspec *d* will be ignored.

SUGGESTED PROJECT

To gain more experience in writing and reading structured files, make a new file and type in the management structure of your department or organization; that is, enter the hierarchy of managers and non-managers. Use level clipping viewspecs to see only personnel at the top level of the hierarchy, then only those at the top two levels, and so on.

SUMMARY

The ability to arrange the information in your MiniBASE file into a hierarchical structure provides several important advantages for writing and reading. Working with structured files not only gives you new tools to help you write; it also changes the way you write. When composing a document in MiniBASE, you are encouraged to plan the overall structure of the document, and this often means that a document written in MiniBASE is more carefully organized than one written by hand or on a typewriter. Further, you can use level and line clipping viewspecs to see the organization of your document at a glance.

This lesson has taught you the commands you need to write a hierarchically structured file. You have learned how to enter statements in insert mode and how to adjust their levels at the "L:" prompt. In addition, you have learned some of the basic viewspecs that make working with and reading structured files much easier.

When you learn to use editing commands to change a structured file, you will see that it is very easy to reorganize a well-structured document; with one command you can move a chapter from the beginning to the end of a document or delete a section of unnecessary or redundant information, without disturbing the rest of the text. You will also find that a document with a consistent structure is much easier to format for printing once you have moved it to AUGMENT, which of course provides much greater flexibility than can a personal computer-based product like MiniBASE.

LIST OF COMMANDS

Insert Statement (to follow) MARK/ADDRESS LEVADJ CONTENT <OK>

<INS> LEVADJ CONTENT LEVADJ CONTENT ... <CD>

<INS> LEVADJ CONTENT TYPEIN ... <CD>

Jump (to) <MARK> VIEWSPECS

Jump (to) Origin (of file) MARK/ADDRESS VIEWSPECS

Jump (to) Next (from) MARK/ADDRESS VIEWSPECS

Jump (to) Back (from) MARK/ADDRESS VIEWSPECS

Jump (to) Successor (of) MARK/ADDRESS VIEWSPECS

Jump (to) Predecessor (of) MARK/ADDRESS VIEWSPECS

Jump (to) Return OK:

<>Set Viewspecs VIEWSPECS OK:

Definitions

MARK Prompted by "M:" or "M/A:" For M you may <MARK>.

TYPEIN Type a series of characters, ending with <OK>. In the Insert Statement command, you may end with <INS> and proceed as shown for <INS> above.

LEVADJ Type any number of level-adjustment characters (u for up, d, <SP> for same level).

CONTENT Prompted by "M/T:" For M you may <MARK>. For T you may type a series of characters, ending with <OK> (if another <OK> follows, a second one is not needed). In the Insert Statement command, you may type <INS> in place of the final <OK> and proceed as shown for <INS>.

VIEWSPECS Type any viewspec characters, ending with <OK>, or just <OK> for no view change.

VOCABULARY

back: In the context of the Jump command, the statement "back" from a statement you indicate is the statement immediately preceding it, regardless of level.

current statement: The last statement you inserted, edited, or jumped to in a file.

down: A term used to describe the relationships between statements at different levels: Statements at level 2 are "down" one level from statements at level 1, statements at level 3 are down one level from statements at level 2, and so on.

hierarchical structure: The structure of MiniBASE files; an outline form that shows the relationships between the statements.

<INS>: "INS" stands for "insert". Typing <INS> puts you in insert mode.

insert mode: You enter this mode when you type <INS>, either at "BASE C:" or in place of the final <OK> at the end of the Insert Statement command. In insert mode, you can continually add statements, each one following the last, until you type <CD>.

Insert Statement command: A command that lets you add statements to a file.

Jump command: A MiniBASE command to move from one point in a file to another, or from one file to another. The statement you jump to is displayed at the top of the file window.

level: A number that indicates how far down a statement is in the hierarchical structure of a file. The greater the number, the further down the statement is in the hierarchy.

level clipping: Using viewspecs to show statements at a limited number of levels.

line clipping: Using viewspecs to show a limited number of lines of each statement displayed.

mark: To mark means to indicate a character on the screen by pointing to it and then typing <OK>.

next: In the context of the Jump command, the "next" statement from a statement you indicate is the statement immediately following it, regardless of level.

predecessor: The predecessor of a statement is the preceding statement that is at the same level and has the same upstatement.

structure: The arrangement of statements in a file. MiniBASE files have a hierarchical structure.

substatement: The substatements of a particular statement are all the statements that are subordinate to it and one level down from it.

substructure: A statement's "substructure" is all the statements subordinate to it, regardless of how many levels they are down from it.

successor: The successor of a statement is the next statement that is at the same level and has the same upstatement.

up: A term used to describe the relationships between statements at different levels: Statements at level 1 are "up" one level from statements at level 2, statements at level 2 are up one level from statements at level 3, and so on.

upstatement: The upstatement of a statement is the statement it is subordinate to and one level down from.

viewspecs: Single-letter specifications of how you see your file. For example, with one viewspec you will see blank lines between statements, and with another you will see the statements without blank lines between them.

REFERENCE

Introduction

This section of the manual assumes that you already have a working knowledge of MiniBASE, AUGMENT, or both (including AUGMENT/MiniBASE's hierarchical file structure), and simply need to answer some questions or need help on a specific point. If you are not familiar with either MiniBASE or AUGMENT, we suggest you turn to the *Tutorial* section of the manual and follow the examples presented there, or else run the interactive tutorials included on your MiniBASE disk.

To use the disk-based interactive tutorials, insert your working copy in drive A and type "tutor<CR>".

Getting Started

Before using MiniBASE at all, you should make a working copy of your MiniBASE diskette and store the original in a safe place. (If you have a hard disk and have installed MiniBASE on it you do not need a backup copy; just store the MiniBASE disk in a safe place). Complete instructions on making a backup (or "working") copy are in the section above ("INSTALLATION INSTRUCTIONS," page 4).

To begin using MiniBASE, you should first either have installed MiniBASE on your hard disk or placed your working copy of MiniBASE in a disk drive. Next, make sure that the DOS prompt shows the letter of the drive from which you will run MiniBASE. If it's on a hard disk, the prompt will probably be "DOS C:"; if you're running from a floppy disk, it might be "DOS A:" or "DOS B:," depending on the drive in which you've placed MiniBASE. If the DOS prompt shows a different drive, you can change it by simply typing the letter of the correct drive followed by a colon and a carriage return.

At the DOS prompt, type "minibase<OK>." You should then see the MiniBASE title screen. This title screen will clear after a few seconds, but if you just can't wait, type any character, and you will immediately be in MiniBASE. Your screen will show "BASE C:" near the upper left corner; the viewspecs "ALL ALL" will be displayed in the viewspec window (upper right-hand corner of the screen). MiniBASE is ready to go to work.

Writing, Creating, Inserting

When you first enter MiniBASE, you see the title screen, and then a nearly empty screen. This is an empty work area, rather like a blank sheet of paper, on which you can begin to work immediately.

To begin a new file:

Create an Origin Statement

To start writing a new document, use the Create File command. Type "<sp>crffilename<OK>", entering for FILENAME the name you wish to give your file. Remember that on a personal computer under PC-DOS, Z-DOS, or MS-DOS, file names are limited to a maximum of eight characters (plus a three-character extension); see your DOS manual for more information on file names.

The Create File command inserts a new statement at the top of your screen (and at the top of your file). This statement has the form of an AUGMENT origin statement. That is, it begins with a left angle-bracket, followed by the filename, followed by a right angle-bracket, followed by four semicolons, and then the date and time the file was created or updated. Following is an illustration of a typical origin statement:

<FILENAME.AUG,>, 25-Apr-85 11:01 ;;;;

MiniBASE creates the origin statement for you, and you should leave it up to MiniBASE to edit at least the portion illustrated above. You should feel free to add text AFTER the four semicolons; but under no circumstances should you edit the text preceding the semicolons. If you do, MiniBASE will probably do you the favor of restoring the origin statement to its original, "correct" form as soon as you update the file -- in the process eliminating the changes you had made.

To work on an existing MiniBASE file:

To load a file from an earlier session, use the Jump (to) File command. Type "jfFILENAME<OK>" and respond to the viewspec prompt by specifying the desired view of the file (or just hit an OK to leave your view the same).

If the file is located on a disk drive other than the "current" drive (generally the one from which you loaded MiniBASE), you will need to type the letter of the drive, followed by a colon, and then the filename. For example, to load the file "PRICES" from the floppy disk in Drive A, you would type "jfa:prices: (for Jump (to) File a:prices).

To work on an existing text file:

To load a file created by some other (non-AUGMENT, non-MiniBASE) editor (e.g., PeachText, Edlin), use the Jump (to) <>Text File command. Type "j<SP>tfilename.EXT<OK>" (don't forget the space between the j and the t), specifying which option MiniBASE should use to break statements (*One*, *Two*, and *Paragraphing)* and respond to the viewspec prompt by specifying the desired view

of the file. (See the section on *Loading an ASCII text file created with other software* on page 75 below for more information on these options.) Remember to include both the filename and the extension (generally .TXT or .DOC); if the extension is not included, MiniBASE will assume it's a MiniBASE (.AUG) file, and the results may surprise you (though no damage to the file will occur).

Please note that such a file is NOT an AUGMENT-style file. Its structure will probably bear little or no resemblance to that of an AUGMENT or MiniBASE file, and its appearance may be different from what you expect. For example, Wordstar files will generally be unusable in MiniBASE. You'll be able to load them as described above, but the screen will be "contaminated" with many strange characters; these are typically special symbols used by the "foreign" editor for formatting purposes.

To work on a file in another DOS directory:

If your computer is equipped with a hard disk, you may find it useful to use different DOS directories to organize your files. If you *do* use multiple DOS directories (DOS 2.0 or higher only), you may want to work on a file in a directory other than the one from which you loaded MiniBASE. To do this, you must first switch to the other directory, and then load the file.

To change to another directory:

Use the *Change Directory* command. Type "<>cd" (don't forget the space!), followed by the name of the directory in which the file is located; then load the file via the Jump (to) File command as described above.

To add information to the file:

Use the *Insert Statement* command. Type "is" and mark the location at which you want the new statement to be inserted. Respond to the level-adjust prompt ("L:") by <OK> to place the new statement at the same level as the one it follows (or you can ignore the level prompt and simply begin typing), or by "u" or "d" to place it a level up or a level down. You may place the new statement more than one level "up" by using the appropriate number of "u"s.

After responding to the level-adjust prompt, you can either begin typing the text of your statement or mark any statement visible on the screen to insert a copy of that statement at the insertion point. Don't worry about using carriage returns to end lines; MiniBASE will adjust your text to fit perfectly without breaking words or wrapping around.

The *INSERT* Mode: Longer Insertions

When you want to insert several statements, one right after the other, you can use the "Insert" key (see the Keyboard Equivalency table on page 3). In insert mode you do not have to give the Insert Statement command over and over; instead you just type your statements one after another, adjusting the level when necessary, and the statements are inserted into your file in the order you type them.

You can type <INS> at "BASE C:" after you have given an Insert Statement command; MiniBASE will assume that you want to begin inserting right after the statement you just entered. When you type <INS>, you see an "L:" prompt. This is the same prompt as in the Insert Statement command, and you should respond to it in the same way: Adjust the level of your new statement as necessary, type the statement, and end with <OK>. MiniBASE will insert the statement into your file and prompt you again with "L:". You can then enter your next statement to follow the one just inserted, adjusting the level as necessary. Follow this with the next statement, and so on, until you have entered all the statements. When you are finished, type <CD> at the "L:" prompt in order to exit the Insert mode.

While it is perhaps safest to enter insert mode after giving an Insert Statement command, you can, in fact, type <INS> any time you are prompted with "BASE C:". When you type <INS>, MiniBASE will assume you want to begin inserting after your current statement, that is, the last statement you inserted, edited, or jumped to. If you are inserting into a newly created file, or just beginning to work, the origin statement will be your current statement and the first statement you insert will follow the origin statement. Because it may be difficult at times to determine just what your current statement is, it is a good practice to begin inserting with the Insert Statement command. After the Insert Statement command, when you know where you are, you can safely enter insert mode.

You can also type <INS> in place of the final <OK> in an Insert Statement command. Because you are replacing <OK> with <INS>, you will not see the exclamation point (!) that lets you know MiniBASE has received an <OK>. MiniBASE will carry out the Insert Statement command as usual and then give you an "L:" prompt just as if you had typed <INS> at "BASE C:". Type <OK> at the end of each statement you insert, and type <CD> after the "L:" prompt when you are done.

Copying Information: the *COPY* Command

You can also add new information or structures to your file by using the *Copy* command (described below, under "Editing").

Editing your work

All MiniBASE editing commands follow the same basic verb/noun syntax used in AUGMENT, so if you're already familiar with AUGMENT you'll feel right at home with MiniBASE. This section is a brief overview of the basic editing operations (verbs) and structures (nouns).

Editing Operations

All five of the logically possible basic editing operations are available in MiniBASE. You can (1) *Insert* new information; (2) you can *Copy* existing information; (3) you can *Delete* information; (4) you can *Replace* information; and you can (5) *Move* it. The operation of these commands is described below, under "Command Summary."

Structures (Things You Can Edit)

There are two kinds of entities or objects you can edit: *strings* and *structures*. Strings (Character, Word, Visible, Invisible, and Text) are contiguous blocks of characters; structures (Statement, Group, Branch, and Plex) are designated portions of files.

Each of the five editing command verbs can be combined with one any of the nine nouns listed above to form a complete command. You can *edit* a Character, a Word, a Visible, an Invisible, Text, a Statement, a Group, a Branch, or a Plex. Each noun is defined briefly below.

STRING Entities

Character: the smallest usable element in a MiniBASE file. Characters can be visible (letters, numbers, punctuation marks) or invisible (spaces, tabs, or carriage returns).

Word: a continuous string of letters and/or numbers. Pointing to ("marking") any character within the word will serve to select the word for editing.

Visible: a continuous string of printing characters (which excludes, for example, spaces and carriage returns). Pointing to ("marking") any character within the visible will serve to select the entire visible.

Invisible: a continuous string of non-printing characters (spaces, tabs, and carriage returns). Pointing to ("marking") anywhere within the invisible will serve to select the entire invisible.

Text: any continuous string of characters (whether visible or invisible) within a statement. To select Text for editing, you must mark both the beginning and the ending characters.

STRUCTURE Entities

Statement: A statement is the basic unit of AUGMENT file structure. A statement can be from 1 to 2000 characters long, and can be a line, a sentence, a heading, a paragraph, or a table -- anything you feel is a logical unit of thought. In a list, for example, each item on the list could be a separate statement only a few words long. If you are writing a book, however, you would make each paragraph a separate statement. You select a statement by marking any character within it.

Group: A group consists of a series of consecutive statements at the same level and all their substructure. In the illustration below, the group defined by 3B and 3C consists of branches 3B and 3C (i.e., statements 3B, 3C, 3C1, and 3C1A). To select a group, you mark the first and last statement in the group. See also: plex.

Branch: A statement plus all its substatements, all their substatements, and so on to the end. The substatements of a branch are sometimes called its "substructure". Even if a statement has no substructure, it still defines a branch. The branch is small, however, and consists of a single statement. In the illustration below, the branch beginning at statement 1 consists of 1, 1A, 1B, 1B1, 1B2, 1B3; the branch at statement 3C consists of 3C, 3C1, 3C1A; the branch at statement 2 consists of statement 2. The branch at statement 0, the branch defined by the origin statement, consists of the entire file. This is true for any file. To select a branch, mark the statement that is the first or top statement in the branch.

Plex: a specified branch, plus all the other branches having the same source. A plex consists of all the branches which have the same upstatement. To select a plex, mark one statement in the plex. In the illustration below, if you asked for a plex and marked statement 3A the entire plex would consist of branches 3A, 3B, and 3C (i.e. statements 3A, 3B, 3C, 3C1, and 3C1A). If you pointed to statement 3B or statement 3C, you would get exactly the same plex as if you pointed to statement 3A.

Illustration of File Structure

Each group of three dots in the illustration represents a statement in the structure of the file. The statement numbers (i.e., "1B1" and its following space) are not part of the statement; they only identify its location. In parentheses are notes on the structural relationships among the statements.

```
0 ... (origin statement)
1 ... (first-level statement, predecessor to 2)
   1A ... (second-level statement, substatement to 1)
   1B ... (second-level statement, back from 1B1)
      1B1 ... (third-level statement, substatement to 1B) }plex, head
      1B2 ... (third-level statement, substatement to 1B) }plex
      1B3 ... (third-level statement, substatement to 1B) }plex, tail
2 ... (first-level statement, successor to 1)
3 ... (first-level statement, upstatement to 3A)
   3A ... (second-level statement, next from 3)
   3B ... (second-level statement)          }branch starting at 3B
      3B1 ... (third-level statement)       }branch starting at 3B
            3B1A ... (fourth-level statement) }branch starting at 3B
```

Moving Around in Files: The Jump Command

The Jump commands move you from one file to another or from one place in a file to another. Jumping also gives you the opportunity to change your viewspecs. (If you're not too sure what viewspecs are, refer back to page 47.) The statement you jump to will be displayed at the top of your file window, with as many following statements as will fit.

To make effective use of several of the Jump commands, you'll need to understand MiniBASE / AUGMENT file structure. If you feel sort of insecure on this topic, you might want to review the "FILE STRUCTURE" section of the Tutorial (above, page 12). You may also find it helpful to refer back to the *Illustration of File Structure* on page 65, to which the examples in the command descriptions which follow refer.

The simplest case of the Jump command involves moving to another location which is visible on the screen. To do this, you simply issue the Jump command and use the cursor to mark the material to which you wish to jump. Type "j" and mark a statement; upon confirmation of the command, MiniBASE will move the statement you marked to the top of the screen, and will of course drag as much of the following material as will fit on the screen after it.

Jump (to) MARK/ADDRESS VIEWSPECS OK:

Jumping to the Top of the File: The Jump (to) Origin Command

The *Jump (to) Origin* command moves you to the origin statement (first statement) of the file. If you want to change views as you jump, when you are prompted for VIEWSPECS, give the new viewspecs you want. If you want the same view, type <OK> for VIEWSPECS.

Jump (to) Origin (of file) MARK/ADDRESS VIEWSPECS OK:

Ignore the "ADDRESS" part of the prompt for now; we'll get to it a little later.

Moving Sequentially through the File

To move to the next statement in the file, whether or not it is visible on the screen, use the *Jump (to) Next* command, which moves you to the statement immediately following the statement you mark. No attention is paid to the relative levels of the statements. If you want to change views as you jump, when you are prompted for VIEWSPECS, give the new viewspecs you want. If you want the same view, type <OK> for VIEWSPECS. See Jump Back (the opposite of Jump Next). See also: Jump Successor.

Jump (to) <>Next MARK/ADDRESS VIEWSPECS OK:

The *Jump (to) Back* command moves you to the statement immediately preceding the statement you mark. No attention is paid to the relative levels of the statements. If you want to change views as you jump, give the new viewspecs you want for VIEWSPECS. If you want the same view, simply type <OK> for VIEWSPECS. See Jump Next (the opposite of Jump Back). See also: Jump Predecessor.

Jump (to) Back MARK/ADDRESS VIEWSPECS OK:

Moving Farther: Jumping by Structure

The *Jump (to) Successor* command moves you to the statement that follows the one you mark, that is at the same level, and that has the same upstatement. In the illustration on page 65, statement 2 is the successor of statement 1, and statement 1b3 is the successor of statement 1b2. If you want to change views as you jump, when you are prompted for VIEWSPECS, give the new viewspecs you want. If you want the same view, type <OK> for VIEWSPECS. See Jump Predecessor (the opposite of Jump Successor).

Jump (to) Successor MARK/ADDRESS VIEWSPECS OK:

The *Jump (to) Predecessor* command moves you to the predecessor of the statement you mark. A statement's predecessor is the statement preceding it at the same level and having the same upstatement. In the illustration on page 65, statement 1 is the predecessor of statement 2, and statement 1b2 is the predecessor of statement 1b3. If you want to change views as you jump, when you are prompted for VIEWSPECS, give the new viewspecs you want. If you want the same view, type <OK> for VIEWSPECS. See Jump Successor (the opposite of Jump Predecessor). See also: Jump Back.

Jump (to) Predecessor MARK/ADDRESS VIEWSPECS OK:

The *Jump (to) Up* command moves you to the statement that is one level higher and comes before the statement you mark; that is, it moves you to the statement's upstatement. In the illustration on page 65, statement 1 is the upstatement of both statements 1a and 1b; statement 1b is the upstatement of statements 1b1, 1b2, and 1b3. If you want to change views as you jump, when you are prompted for VIEWSPECS, give the new viewspecs you want. If you want the same view, type <OK> for VIEWSPECS. See Jump Down (the opposite of Jump Back).

Jump (to) Up MARK/ADDRESS VIEWSPECS OK:

The *Jump (to) Down* command moves you to the statement one level lower and following the statement you mark. In the illustration (page 65), statement 1b1 is the substatement (or "Down") of statement 1b, and statement 3a is the substatement of statement 3. *[Note that Jump (to) Down will be equivalent to Jump (to) Next -- if and only if the statement you mark has a substatement.]* If you wish to change your view as you jump, enter new viewspecs at the end of the command, for VIEWSPECS. To keep the same view, simply type <OK>. See Jump Up (the opposite of Jump Down). See also: Jump Next.

Jump (to) Down MARK/ADDRESS VIEWSPECS OK:

The *Jump (to) Head* command moves you to the first statement in the plex you mark. In the illustration on page 65, statement 1 is the head of the plex at 2 (or 3); statement 1a is the head of the plex at 1b; and statement 1b1 is the head of the plex at 1b2 or 1b3. If the statement you indicate is the only statement or the last statement in the plex, you will be taken to that statement. If you want to change views as you jump, when you are prompted for VIEWSPECS, give the new viewspecs you want. If you want the same view, type <OK> for VIEWSPECS.

Jump (to) Head MARK/ADDRESS VIEWSPECS OK:

The *Jump (to) Tail* command moves you to the last statement of the plex you mark. In the illustration (page 65), statement 1b3 is the tail of the plex at 1b1 or 1b2; statement 3 is the tail of the plex at 1 or 2. If you want to change views as you jump, when you are prompted for VIEWSPECS, give the new viewspecs you want. If you want the same view, type <OK> for VIEWSPECS. See Jump Head (the opposite of Jump Tail).

Jump (to) Tail MARK/ADDRESS VIEWSPECS OK:

Jumping Back in Time: The Jump Return Command

The *Jump Return* command allows you to move back to in-file locations in which you were working recently. MiniBASE stores the last ten locations you've visited

in your current file, and allows you to return to them with the *Jump Return* command.

Jump (to) Return OK:

MiniBASE will respond by presenting the first few words of the last location. Answering "y" to the "Y/N:" prompt will return you to that location. If you respond with "n" for No, you'll be offered another location (one location back in time from the one you just declined), and can say yes or no to that one -- and so on, back through a "return ring" of 10 locations.

Jumping to a Specific Screen Location

Occasionally -- especially if you happen to be looking at a statement which is too long to fit entirely on the screen -- you may want to move the middle of a statement to the top of the screen. To do this, use the *Jump (to) Character* command and mark the character you would like to move to the top of the screen. MiniBASE will make that character the first character of the first line on the display, and will fill the rest of the screen with as much of the following material as will fit.

Jump (to) <>Character MARK/ADDRESS VIEWSPECS OK:

Jumping to Locations Not on the Screen: Using Addresses

Sometimes the location to which you want to Jump isn't on the screen. Of course, you could keep jumping to the "Next" statement, or to the "Successor," to keep scrolling down through the file. You might even remember your viewspecs, and use viewspec "t" (one line only) so as to get more statements on the screen at a time.

But there is an easier, faster way (actually, two ways; we'll cover the second in the next section). Each statement in your file has a statement number associated with it, and you can see those numbers or not, depending on what you want. Like the blank lines between statements, the numbers are turned on and off with a viewspec. Viewspec "m" turns on the numbers, and viewspec "n" turns them off.

Statement numbering reflects the organization of the information in your file in a manner corresponding to standard outline structure. The highest-level statements (e.g., chapter titles) are numbered sequentially (1, 2, 3....). Lower-level, subordinate sections are labelled with a combination of numbers and letters (e.g., 1a, 1b; 2a4, 2a5), with the number structured to precisely reflect the position of the statement in the file.

So if you were working in the middle of the first section of your file (the branch headed by statement 1), and you wanted to jump to (for example) the fourth

subsection of the third major section of your file, you could simply Jump (to) Item 3d!

Of course, it's easiest to find a statement if you already know its number -- you can't always remember exactly where it is in the file. Which is why some people, when they print out a draft of their work, do so with viewspec "m" in effect. This prints the statement numbers in front of their statements, which can greatly simplify subsequent editing.

Actually, the Jump command isn't the only one that uses addresses. In fact, any time you see the "M/A:" prompt, you can enter a statement number ADDRESS instead of MARKing a statement. This allows you to edit statements that aren't on the screen (sometimes dangerous, but definitely useful, particularly when you're working with large portions of a file).

Sometimes, however, you haven't printed your file, and you can't remember which section (exactly) your statement was in -- only that it had something to do with "third-quarter budgets." What then?

Searching for a Specific Word or Phrase

The *Jump (to) Content* commands allow you to specify one or more characters and to jump to either the first or the "next" occurrence of the character(s) in the file. Use *Jump (to) Content First* to go to the first place in the entire file where the character(s) occur. Use *Jump (to) Content Next* to reach their first occurrence after your current location. When the characters are found and the jump is completed, the statement containing the character(s) will be at the top of the file window. Below this will be as much of the following statements as will fit on the screen.

Jump (to) Content First MARK/TYPEIN VIEWSPECS OK:

> The *Jump (to) Content First* command takes you to the first appearance in your file of the character(s) you specify. You can either TYPEIN or MARK to specify the character(s). For VIEWSPECS, you can give viewspec codes to change your view when the jump is complete; to keep the same viewspecs, simply type <OK>. MiniBASE looks for character(s) capitalized exactly as you specify; be sure to search separately if necessary for all the different types of capitalization. To repeat a content search, use the TAB command (<CTRL-I>).

Jump (to) Content Next MARK/TYPEIN VIEWSPECS OK:

> The *Jump (to) Content Next* command searches your file, from where you are to the end, for the next appearance of the character(s) you specify. You

can either TYPEIN or MARK to specify the character(s). For VIEWSPECS, you can give viewspec codes to change your view when the jump is complete; to keep the same viewspecs, simply type <OK>. MiniBASE looks for character(s) capitalized exactly as you specify; be sure to search separately if necessary for all the different types of capitalization. To repeat a content search, use the TAB command (<CTRL-I>).

Statement Names

Any MiniBASE statement (or any AUGMENT statement, for that matter) can have a name which you can use in the *Jump (to) Item* command. Statement names are always the first visible in the statement, but the first visible is not always a statement name; in order to be a statement name, the first visible must be surrounded by valid *statement name delimiters;*, which in MiniBASE can be any character, including the NULL character (i.e., nothing).

MiniBASE statement name delimiters as set in your configuration file are your default name delimiters. If you have not changed them (i.e., if you are still using the default name delimiters supplied in the MB.MBC configuration file which came on your distribution disk), they are NULL NULL (shown on the *Setup* screen as '^@ ^@'); in this case, the first word (or visible) in each statement will be the name of that statement. For example, you could jump to this statement by saying, 'Jump (to) Item minibase' and confirming the command. (Note that capitalization doesn't matter.)

You can set the name delimiters within a MiniBASE file by placing a Name Delimiter Directive in the origin statement of the file. This directive has the form

```
.NDlim=AB;
```

where A is the left statement name delimiter, and B is the right statement name delimiter. For example, you could set your statement name delimiters within a particular file to parentheses by including the following directive in the origin statement:

```
.NDlim=();
```

Statement names can be turned on and off (i.e., rendered visible and invisible) with viewspecs C and D. If viewspec C is in effect (the default), statement names are visible; with viewspec D, statement names will disappear, but they will still exist in the file and can still be used for addressing in the Jump (to) Item command.

Statement names thus provide a simple, powerful, and flexible form of addressing within MiniBASE. You could, for example, use parentheses as name delimiters, and place the string '(SectionN)' at the beginning of each new section of your

document (with N being the section number). You could then make these section labels invisible by setting viewspec D, and could always jump directly to the beginning of a section, no matter how much you had edited the file, with the Jump (to) Item command. For example, to jump directly to the beginning of section three, you could use the following command:

Jump (to) Item section3<OK><OK>

For more information on statement names and statement name delimiters, see the section on configuration files in the section on the *Setup* program at the end of this chapter.

Changing your View of the File: The Set Viewspecs Command

You can change your view of the file at any point by entering the *Set Viewspecs* command, followed by the viewspecs you want. MiniBASE will recreate the display with the new view of the file.

Set Viewspec VIEWSPECS OK:

Advanced Editing

Global Search and Replace: The Replace All Command

Have you ever mispelled a word consistently throughout an entire document? or needed to change 'PC' to 'personal computer' in a chapter or file? The *Replace All* command is what you need.

The Replace All command looks through the *structure* you specify, and replaces every occurrence of the old text you specify with the new text you specify. (For more information on structure, see above, page 64.)

To use the Replace All command:

Replace All (in) STRUCTURE (at) MARK/ADDRESS (of old text) MARK/TYPEIN (by new text) MARK/TYPEIN OK:

Working with Split Screens

Although most of the time your work will require you to use only one file at a time, there are many times when you need to use two at once. For example, you might want to compare the contents of two different files, or to use one file as a source document in building a second by copying portions of the first.

In a paper-based world you would put the two documents side by side on your desk and go to work. MiniBASE lets you do the same thing on your display screen by using the *Break Window* command (also useful for venting pent-up anger and frustration). By splitting your screen horizontally into two different "windows," MiniBASE gives you two separate, independently controllable work surfaces into which you can load either two different files or two different sections or views of the same document.

To use the command, type "bw" -- you'll see "Break Window (horizontally at) M/T:". MiniBASE is waiting for you to tell it the lower boundary of the upper window, which will contain the current file. You can respond by MARKing the location at which you want the break to occur, or by entering a line number (from 0 to 23, numbering from the top of the screen down), and then a final <OK> to confirm the command. The result will be a screen displaying your current file in the upper window; the bottom window will be empty.

Position your cursor in the lower window, and use the Jump (to) File command to load another file into the lower window. You are now ready to edit either file -- or both at once! (Just for the fun of it, try using the Transpose command (described below) to swap text, or words, or whole statements, across windows. Simultaneous cross-file editing! Sure hope you remembered to update both files first! And then Delete Branch (at) 0 to restore their former glory....)

As we mentioned above, you can also use this split screen capability to display different portions or views of the same file in the two windows. This can be very useful when you're editing a large file -- for example, in moving entire portions of a document from one section to another when you don't know the statement numbers of either the source or the destination.

Returning to Full Screen Mode

When you're through with the split screen, and need your full screen back again, use the Break Window command to restore the full screen by typing "bw23", or "bw" and MARKing the bottom of the screen.

Changing Window Sizes

You can change the relative size of the two windows by using the Break Window command and specifying a different location for the split. MiniBASE will expand one window and contract the other, precisely as you have specified.

Memory Management

When you load a file into MiniBASE via the Jump (to) File command, MiniBASE copies the file from disk into your computer's random access memory (RAM).

MiniBASE can deal with quite a bit of memory all at once -- about 64K, which is actually 65,536 bytes (computerese for characters).

But MiniBASE doesn't care whether that's all in one file, or whether it's in two files loaded into separate windows. The result is that you may not always be able to load a entire file into the second window. For example, if you started a worksession with a 42K file occupying the full screen, and then broke the window into two sections and loaded a 20K file into the bottom window, everything would be fine; but if you tried to load a 30K file into the bottom window, MiniBASE would bump into the 64K limit. It would do the best it could; it would load as much of the second file as the 64K limit would allow, and would alert you by saying, "File buffer full--truncated."

Once the "file buffer" is full, you will be unable to add anything to either file until you have removed something from the buffer, either by using the Delete command to remove a portion of one file or the other from memory, or by using the Delete Branch (at) 0 command to remove the file in the bottom window from memory.

Once you have used the Break Window command in a MiniBASE worksession, MiniBASE will "remember" the file you had in the lower window (i.e., the file remains in the "file buffer"), so that if you later issue the Break Window command, you'll find the lower window occupied by the same file it held earlier. Furthermore, remember that such a file continues to occupy space in your computer's RAM, and thus could conceivably cause you to bump into the 64K memory limitation we were just discussing, even though it's no longer on the screen; thus, it's a good idea to remove the file completely from memory via "Delete Branch (at) 0" when you're sure you're through with it. After all, you can always get it back.

Splitting and Joining Statements: the Break and Append Commands

When you need to separate a single long statement into one or more shorter statements, use the *Break Statement* command. Type "bs" and MARK the "visible" (any continuous string of printing characters, but *not* including spaces, tabs, or carriage returns) you would like to be at the *end* of the first statement. You have the option of specifying a different level for the new (second) statement.

To join two separate statements together, use the *Append Statement* command. Type "as" and either mark the statement you want to be at the end in the new statement to be formed, or enter its ADDRESS (statement number); then respond to the "(to)" prompt by MARKing (or giving the address of) the statement that's to come first. You will be prompted by "(join with)"; enter whatever text you'd like to be inserted between the two statements when they're glued together. (If you

enter nothing in response to the "(join with)" prompt, you will get nothing in between -- you'll usually wish you had at least put in a space or two.)

Swapping Text: The Transpose Command

Transpose is one of the most unusual commands in MiniBASE. It allows you to exchange any two of the many editable objects (e.g., characters, words, statements, branches....), even across files. Simply type "t" (for Transpose), tell MiniBASE what kind of object it is (use the question-mark if you're not sure what's available), and then MARK or ADDRESS the two objects; they'll trade places immediately.

Saving your work

MiniBASE keeps a working copy of the file on which you're working in your computer's RAM (Random Access Memory). RAM is what is called "volatile" memory -- that is, information stored there will disappear when the computer is turned off -- whether by you or by a power failure. In order to save your work for future use, use the *Update* command to store a copy of the current file (the one in RAM) on disk. We suggest that you do this fairly often (every ten or fifteen minutes) in order to protect yourself from losing much work in the event of a power failure.

If you are working on a new file:

If you took our advice and used the Create File command to set up a "proper" origin statement at the top of the file, MiniBASE already knows what you've chosen to name your file, so you can simply use the *Update File* command. Type "uf<OK>".

If you haven't created an origin statement for your file, you can do so now, and then use the Update File command as just described. The origin statement will contain the text of the first statement you inserted in the file, preceded by the standard origin statement format (i.e., FILENAME.AUG,>, 22-Jan-86 10:05 ;;;;).

If you prefer *not* to have an AUGMENT-style origin statement, you can still use the Update File FILENAME command, entering for FILENAME the name (up to eight characters) under which you'd like your file to be stored. You can use any alphabetic or numeric characters, and some other "special" characters as well. Consult your DOS manual for more information on file naming; MiniBASE is required to follow the same rules.

Type "ufFILENAME<OK>". MiniBASE will create a new file named FILENAME.AUG on the current disk drive. (To save the file on a different disk drive, preface the filename with the letter of the disk drive and a colon -- e.g.,

"A:FILENAME". MiniBASE does not currently support the use of DOS pathnames.).

If you are working on a file from an earlier session:

Use the Update File command. Type "uf<OK>". MiniBASE will create a new version of the original file (FILENAME.AUG) on the same disk drive from which the file was loaded. The new version will contain all the changes you've made to the file. In addition, a backup copy of the original file will be retained in a file called FILENAME.AU@.

To save your current work under a new file name:

Use the Update File FILENAME command as described above, entering for FILENAME the new name you'd like to give to the revised file. MiniBASE will save your work under the new file FILENAME.AUG. Your original file will still be there too under its old name.

To create a text file for use with other software:

Occasionally you might want to use MiniBASE to create a file for input to other software. For example, you might want to use MiniBASE to set up a BASIC program, or to provide input to a typesetting program. Because other software does not understand MiniBASE / AUGMENT's hierarchical file structure, you will need to create an equivalent text file. Use the Update Text File command. Type "u<SP>tFILENAME.TXT".

Loading an ASCII text file created with other software:

MiniBASE can be used to edit files created with other software, or to import ASCII text output from programs such as Lotus or DBASE III[1]. Use the *Jump Text File* command to load the file:

> Jump (to) <>Text File FILENAME.TXT (using) One/Two/Paragraphing VIEWSPECS OK

The three options (*One*, *Two*, and *Paragraphing*) after the filename tell MiniBASE how to figure out where one statement ends and another one starts.

One (carriage return to end statements): Selecting this option tells MiniBASE to start a new statement each time it encounters a carriage return <CR> character in the text file. Generally, this will cause each line in the text file to be brought into MiniBASE as a separate statement; the exception is any program which formats the screen "on-the-fly" instead of ending lines with an actual <CR> character.

One (carriage return....) is the recommended option for things like tables -- for example, the formatted print file produced by Lotus™.

Two (carriage returns to end statements): MiniBASE will start a new statement everywhere it sees two <CR> characters in a row. This option is particularly well-suited for files created in PC-based word processing packages (e.g., MultiMate™) and saved in a 'non-document' or 'text only' mode. Following are some basic rules for setting up such files for easiest transfer to MiniBASE:

Files created in other word processing packages can generally be imported into MINIBASE if the following simple rules are followed in creating and editing the files.

1. *No formatting in the PC file.* Use the PC software only for writing and editing. PC editing tools often use special formatting characters for things like underlining, centering, and bolding. If you upload a file with such characters in it, MINIBASE won't know what to do with them, and they'll appear in your MINIBASE file as "garbage characters," which you'll then have to edit out. Instead, strip out all underlines, centering codes, boldface, etc., before importing the file into MiniBASE.

2. *Avoid the use of tabs.* Use multiple spaces instead.

3. *Double-space between paragraphs, single-space within.* MiniBASE will use spacing to locate statement breaks.

4. *Use indentation to reflect structure.* That is, make your DOS word processing file look like what it would be if it were a MINIBASE file -- indent each statement/paragraph three spaces per level. Only the first line of each paragraph (statement) must be indented; indent the subsequent lines or not, whichever is more convenient for you. Indentation of all lines by the same amount as the first can make it easier for you to see the "structure" of your DOS file, thereby making it easier for you to follow this rule consistently.

Paragraphing: Selection of this option tells MiniBASE to decide how best to break statements. MiniBASE will try to put paragraphs into separate statements, assigning levels according to the indenting of the paragraph as a whole. It uses the indenting of both the first and the second lines of the paragraph. It also looks at short lines -- ones with enough space at the end for words from the next line -- as an indication of a paragraph end.

To get rid of the changes you've made in this session:

It happens to all of us. You've been working for half an hour on modifying a document and are unhappy with what you've done, and would do almost anything to simply scrap the changes you've made and get back to the original. Here's what to do (have faith and give it a try! it works, so long as you haven't updated the file):

Get back to the top of the file. Use the Jump (to) Origin command.

Delete Branch (at) (MARK the origin statement).

Or you can simply type "db0<OK><OK>".

Your screen should now be empty except for "BASE C:" and the viewspec display. You can either Quit or use the Jump command to load a file -- including the original, unmodified version of the file you were just working on.

Working with files

The current version of MiniBASE provides minimal file manipulation capabilities. You can list your files with the Show Directory command (type <>Show Directory<OK>); future releases will allow you to list your files, and to delete, rename, or copy them; for now, though, you'll need to perform these operations from DOS instead of from MiniBASE. (You can use the Goto DOS command to go to the DOS level to use these commands and then return to exactly where you left off in MiniBASE. See below, page 80).

The following DOS commands are useful for file manipulation. See your DOS manual for a full explanation of these commands.

DIR (Directory) Command

The DOS DIR command lists the files on a designated disk, or in a designated DOS directory. Type "dir<OK>" for a full listing. It can also provide a list of all files having certain characteristics -- for example, all MiniBASE files (files having the extension "AUG" or "AU@") could be listed by the following DOS command:

```
DIR *.AU?<OK>
```

The asterisk and the question mark are DOS "wild cards;" their use is documented in the DOS manual.

COPY Command

The DOS Copy command copies files from one disk to another, or to the same disk but with a different name.

> COPY OLDFILE.AUG NEWFILE.AUG<OK> -- creates a copy of OLDFILE in a second file named NEWFILE.
>
> COPY OLDFILE.* B: -- copies all files named OLDFILE on the current drive, regardless of their extension, to the disk in drive B.

DELETE (DEL) Command

Removes the specified file from storage.

> DEL FILENAME.AUG -- deletes the MiniBASE file FILENAME.AUG from the current drive.
>
> DEL TEMP.* -- erases all files named TEMP, regardless of their extension, on the current drive.
>
> DEL A:*.AU@ -- deletes all the MiniBASE backup files (files with the extension AU@) from the disk in Drive A.

RENAME (REN) command

Changes the name of a file.

> REN OLDFILE.AUG NEWFILE.AUG<OK> -- renames the file OLDFILE.AUG to be called NEWFILE.AUG. (The origin statement of the new file will still contain the name of the old file; it will change to NEWFILE the first time you update it.)

Printing and Formatting: The Print Command

MiniBASE was designed to provide basic hardcopy output capabilities, to allow you to produce drafts of large documents (up to MiniBASE's maximum capacity of 64K) and final copies of short documents (e.g., letters and memos). Use the Print command (type "p" for Print, followed by B(ranch), P(lex), G(roup), or S(tatement). If you need extensive printing or output formatting capabilities, you will need to use AUGMENT. Your Augterm User's Guide gives full information on moving files back and forth between your workstation and AUGMENT, while the Output Processor User's Guide provides complete information on AUGMENT's formatting capabilities; ask your McDonnell Douglas representative for help in learning more.

The Print command

The MiniBASE Print command is easy to use. Each person has a personalized set of "configuration files" which tell MiniBASE exactly how he or she usually wishes to print. Thus every user could have a different type of printer, but all would use the same printing commands. These configuration files are maintained and modified using the *Setup* program, which is on your distribution disk. (A basic set of configuration files has been included on your MiniBASE diskette.) See the section on *Setup* below for more information.

For now, let us assume that all your configuration files are properly set up, and that you wish to print in your "usual" way. All you need to do is type "p" for "Print" and then identify exactly what you wish to print -- a statement, branch, group, or plex.

Remember that the Print command obeys the viewspecs currently in force. If you had viewspec *x* in effect, so you were viewing only the first lines of the top-level statements, that's all that would be printed -- so make sure your viewspecs are what you want *before* you use the Print command.

Using AUGMENT

Although the terminal emulation and communications software (*Augterm*) needed for using AUGMENT is a separate program, you can call it from MiniBASE. This means that you can go to AUGMENT to check your mail, or to upload or download a file, or anything else -- all without Quitting from MiniBASE. In other words, you can simply go to AUGMENT, do what you need to do there, and then come back to MiniBASE to resume whatever you were doing before you went to AUGMENT.

Note: This capability requires that your personal compuyter be equipped with a minimum of 256K of RAM and DOS 2.0 or higher. It cannot be used on a 192K Z-100 under Z-DOS.

The command to do this is *Goto AUGMENT*. Type "ga<OK>" -- the screen will go blank, at which point you're ready to dial up and enter AUGMENT.

Consult your *Augterm* manual for instructions on using *Augterm*.

Goto Augment<OK>

Using DOS Commands or Other Software

DOS Commands

There are many times when you might need to use a DOS command -- for example, to copy, delete, or rename a file -- and would prefer to do so without Quitting from MiniBASE and closing your file. Using the *Goto DOS* command allows you to do this. Simply type "gd<OK>"; the DOS prompt will appear at the bottom of the screen, and your MiniBASE window will scroll up to make room for it. You are then free to use any DOS commands -- or indeed, any other software (subject to the memory available).

When you're finished with DOS and want to return to MiniBASE, type "exit<CR>".

Goto DOS<OK>

Other Software

You can run other software as described above, by using the Goto DOS command and then running the software at the DOS prompt (assuming you have enough memory in your computer to have both MiniBASE and your other software in memory simultaneously). You can also run it directly from MiniBASE with the *Goto <OPT>Program* command.

Goto <OPT>TYPEIN<OK>

For TYPEIN specify the full name of the program you wish to run, including its extension. For example, to run DBASE III™ you would type "g<OPT>dbase.exe<CR>". *(Note: the program you wish to run must be in your current DOS directory. If you are running MiniBASE from a hard disk, you probably have placed MiniBASE and your other AUGMENT-related files in their own directory. To use DBASE, you would need to switch to the DBASE directory. Use the MiniBASE "Change Directory" command to do this -- see above, page 61.)*

Ending the session

When you are finished working with MiniBASE, make sure you've updated the file on which you were working, and then use the Quit command to leave MiniBASE. Type "q<OK>"; you'll return to the DOS prompt.

Note that MiniBASE will not let you Quit while a file is being modified -- that is, until you have updated it. If you want to abandon your modifications to a file, see

the instructions given above under "To get rid of the changes" in the section on "Saving your work."

MiniBASE Configuration Files: the SETUP Program

Introduction to SETUP

Like many personal computer programs, MiniBASE uses configuration files to control the way it works. The appearance of the screen, basic printing parameters, and statement name delimiters are some of the variables included in MiniBASE configuration files.

You don't need to know anything about configuration files to use MiniBASE. We've included one on your distribution disk -- MB.MBC. This file contains basic default settings for all settable parameters, and is automatically loaded along with MiniBASE -- so you've been using it as long as you've used MiniBASE.

You may never need to change any of the settings in MB.MBC. On the other hand, you might want to -- particularly if you have a color monitor, or if you find the default printing parameters inappropriate for your work. If you do, the Setup program is available to help.

Zenith Z-100 users please note: Setup on the Z-100 is somewhat different from the description below, which applies to the version for the IBM PC and compatibles (including the Zenith Z-150 and Z-248). See the section below on 'Z-100 Version' for more information on specific differences.

Instructions for Use

Loading SETUP

From DOS

To start Setup from DOS, simply type SETUP<CR>. After a brief pause the Setup screen will appear with the default configuration file (MB.MBC) loaded.

To run Setup with a different configuration file, enter the filename on the DOS command line after the word 'setup.' Type SETUP FILENAME<CR>, where FILENAME is the name of the .MBC File you want to modify. (If FILENAME.MBC does not exist Setup will create it for you, setting it up with the default values, which you are then free to modify).

From MiniBASE

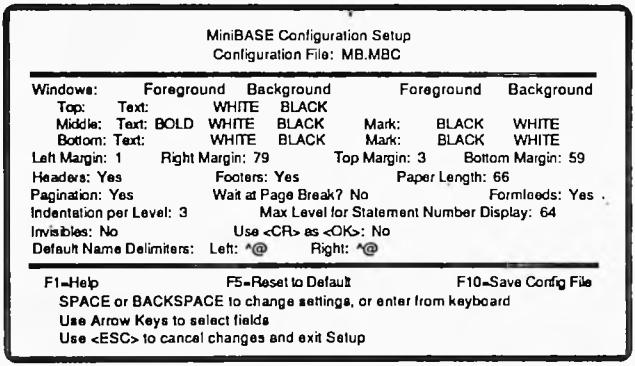
If you're working in MiniBASE and you want to modify your configuration file, use the Goto <OPT>Setup command. Type 'g<OPT>setup<OK>'. Setup will begin, with the default configuration file (MB.MBC) loaded.

To work on a configuration file other than the default (MB.MBC), simply enter the name of the desired file in the Goto command. Type 'g<OPT>setup filename<OK>' and setup will begin with the specified file. If the file does not exist, Setup will create it and will supply the default values, which you can then modify.

NOTE: SETUP.COM must be in the current DOS directory or the root directory in order to be run from MiniBASE. Also note that your personal computer must have enough memory to run MiniBASE, Setup, and any other software (especially RAM-resident, TSR-type programs) you may have loaded. A minimum of 256K is recommended.

Using SETUP

Setup is easy to use. When you load it, a screen similar to the one below will appear:



Simply use the arrow keys to position the cursor on the value you want to change, and then either use the space bar to step through the allowable values (for items like screen color) or enter them directly from the keyboard (for items like margin settings).

If you want to restore a field to its default value, position the cursor in that field and type F5.

To update your configuration file to contain the new values, type F10. You'll see the following prompt:

> Update File D:FILENAME.MBC T/<OK>/<CR>, or <ESC> to abort

Where

> D is the current drive, and
>
> FILENAME is the name of the configuration file you are editing.

To save your changes on the current configuration file, simply type <OK> or <CR>,

To create a new configuration file which contains the changes you've made in this session, enter the name you want to give to the new file, followed by <OK> or <CR>. The new file will be created; your original file will remain unchanged.

To abort and return to Setup, type <ESC>.

To cancel the changes you've made and exit from Setup, leaving your configuration file unchanged, type <ESC>, and then confirm with <OK> or <CR>.

Once you have finished using Setup, you can have MiniBASE use your new configuration file in either of two ways:

From DOS: When you load MiniBASE, type MB FILENAME (where FILENAME is the name of an existing .MBC file). MiniBASE will load and run, using the values it finds in the configuration file you specified.

If you enter the name of a non-existent .MBC file, MiniBASE will so inform you with the message, 'File not found.' MiniBASE will then load and run using default values. You can then use the Set Configuration command to load any of your .MBC files.

If you use the MINIBASE.BAT batch file from your distribution disk to load and run MiniBASE, along with the Mouse Support files, the configuration file used will always be MB.MBC (the default configuration file). To use a different configuration file, use the Set Configuration command in MiniBASE, or edit the batch file to include a different .MBC file.

In MiniBASE: Use the Set Configuration command (see below, page 91).

Settable Configuration Items

Default Name Delimiters

Name delimiters are single characters that enclose and define statement names. A delimiter can be any visible character, or it can be nothing, shown as "^@". Spaces between the delimiters and the name are allowed, and, if the delimiters are valid name characters, spaces between the delimiters and the name are required. The name delimiters used by Setup are NULL on the left and NULL on the right.

Control characters can be entered either as the control character itself, or with the two-character visible sequence as displayed in the field. Some control characters (including ^@) must be entered as a visible sequence.

<ESC> cannot be used to abort Setup when the cursor is in one of the name delimiter fields; it would be understood by Setup as a name delimiter entry. To abort, move the cursor out of the name delimiter fields and then press <ESC> and confirm with <OK> or <CR>.

Note that MiniBASE's implementation of statement names differs from that of AUGMENT. Setting name delimiters via Setup is the MiniBASE equivalent of setting your name delimiters in AUGMENT via the Set Profile command; however, in MiniBASE, name delimiters cannot be changed within a file -- i.e., they apply file-wide. They can be set for a particular file; see the section on Statement Names above (page 70).

Allowable settings: Any character
Default Settings: Left: ^@ Right: ^@
(^@) is Setup's way of displaying the NULL character; i.e., the default statement name delimiters are NOTHING and NOTHING. Thus the first visible in each statement will be the statement name.)
Change via: Keyboard entry.
Affects: Display only.

Formfeeds

When this value is set to Yes, MiniBASE will send a Formfeed character (ctrl-L) at the end of each page to force your printer to eject a new page. If your printer does not respond appropriately to the Formfeed character, set Formfeeds to No; MiniBASE will then send a series of carriage return characters to move to the top of the new page (slower than Formfeeds, but the effect is the same).

Allowable settings: Yes/No
Default Settings: Yes
Change via: Space Bar
Affects: Printed output only.

Headers & Footers

When these values are set to Yes, MiniBASE will print headers and footers on each page of output. The footer will consist of the page number, centered at the bottom of each page; headers have the following form:

```
4-Mar-87 11:22                                    < SAMPLE.AUG, > 1
```

Date and time (from DOS) appear in the upper left-hand corner of each page. In the upper right-hand corner, MiniBASE places the name of the file, followed by the page number.

If both Headers and Footers are on (set to Yes), the page number will appear at both the top and the bottom of each page.

Allowable settings: Yes/No
Default Settings: Yes
Change via: Space Bar.
Affects: Printed output only.

Indentation per level

This setting controls the number of spaces MiniBASE will indent each lower-level statement relative to the statement it follows. You might want to change this setting if you had a file with many levels and you found that the total indentation was becoming excessive.

Allowable settings: 1-20
Default Settings: 3
Change via: Keyboard entry.
Affects: Both display and printed output.

Invisibles

Invisibles -- spaces, tabs, and carriage returns -- are called invisibles for one reason: you can't see them. And that's the way you'd like it to be -- except when you're editing a file in which such characters have been used to control formatting, and you suddenly realize that you can't tell the difference between (for example) a row of spaces and a couple of TAB characters. This setting allows you to make invisibles visible -- not by making any change in your file, but by changing the way such characters are displayed on the screen. Currently, spaces are displayed as spaces, TABs as tiny 'o' characters, and carriage returns as small musical notes. In a future version, spaces will be displayed as tildes (~), TABs as right anglebrackets (>), and carriage returns as left anglebrackets (<)).

Allowable settings: Yes/No Default Settings: No Change via: Space Bar. Affects: Display only.

Margins

The Left, Right, Top, and Bottom margins for printed output are independently settable. Position the cursor on the margin field you want to change and enter the desired value from the keyboard.

Do *not* set your left margin to a value greater than your right margin, or your top margin to a value greater than your bottom margin. You'll be sorry if you do.

Allowable settings: Left Margin: 1-50 Right Margin: 1-160 Top Margin: 0-30 Bottom Margin: 1-132 Default Settings: Left Margin: 1 Right Margin: 79 Top Margin: 3 Bottom Margin: 59 Change via: Keyboard entry. Affects: Printed output only.

Max level for statement number display

This setting controls the maximum level number for which statement numbers will be displayed when viewspec m is used. For example, if this value is set to 5, and

viewspec m is in force, statements below level 5 (i.e., from level 6-64) will not display statement numbers.

Although this setting affects both display and printed output, it is used primarily to control the latter. As you move deeper and deeper into a MiniBASE (or AUGMENT) file, the statement numbers get longer and longer. For example, a level 5 statement might be numbered 5b3d2. Most people find that the ugliness of statement numbers tends to increase in direct proportion to their length; this setting allows you to put a stop to this increase.

Allowable settings: 0-64
Default Settings: 64
Change via: Keyboard entry.
Affects: Both display and printed output.

Pagination

When Pagination is set to Yes, MiniBASE will begin a new page after each N lines, where N=Bottom Margin-Top Margin. Generally this is what you'll want, but if you're printing (for example) program source code on continuous-form paper, you might want to turn Pagination off.

Allowable settings: Yes/No
Default Settings: Yes
Change via: Space Bar.
Affects: Printed output only.

Paper length

Most printers print six lines per inch, and most paper is 11 inches long. We've therefore set the default page length to 66 lines. If your printer spaces lines differently, or if you are printing on some other size paper, you may want to change this value.

Allowable settings: 20-128
Default Settings: 66
Change via: Keyboard entry.
Affects: Printed output only.

Use CR as OK

When this value is set to No (the default), the <CR> key functions simply as a standard character and is used to embed a carriage return character in a file; commands can only be confirmed by the <OK> key. This generally provides more

flexibility by making it easy both to confirm commands and also to place carriage return characters in your files.

On some keyboards, however, the <OK> may be inconveniently placed, as may the F8 key (a synonym for <OK>). In such cases it is more convenient to have the <CR> key function to confirm commands. Setting this value to Yes causes the <CR> key to equal <OK>. If you have set this value to Yes, and you want to enter a carriage return into your file, <Shift-CR> will give a carriage return instead of <OK>.

Allowable settings: Yes/No
Default Settings: No
Change via: Space Bar.
Affects: Display only.

Wait at Page Break?

When this value is set to Yes, MiniBASE will pause at the end of each printed page and prompt you:

Insert new page and type control-G

This allows you to insert a new sheet of paper, or to verify or change page alignment in the printer. If you are printing on continuous-form or fanfold paper you will generally want this value set to No.

Allowable settings: Yes/No
Default Settings: No
Change via: Space Bar
Affects: Printed output only.

Window Colors

The MiniBASE screen is divided into three different areas. Each area may be thought of as a window in which Setup can separately control screen colors.

MiniBASE windows are described on page 13 above. Setup groups these windows as follows:

Top Window -- the top two lines (lines 0-1) of the display, containing the Status Window, the Viewspec Window, and the Command Window.

Middle Window -- The first, uppermost text display area, or File Window. When you enter MiniBASE, this window fills the display below the Top

Window -- i.e., lines 2-23 of the display. If you use the Break Window command, you create a second File Window, which appears below the Middle window on the screen. In Setup, this window is referred to as the Bottom Window.

Bottom Window -- The second, lower text display area, or File Window. This window can be of any size up to the entire text display area (lines 3-24); normally, it fills some portion of the display below the Middle Window. If you use the popup menus from the Mouse Support disk to break your window into two File Windows, the break will occur in the middle of the display, at line 12.

The Foreground and Background colors are independently settable in each of these three windows. In addition, you can also independently control the TEXT and MARK colors in the Middle and Bottom Windows.

Foreground Color -- the color in which text characters will be displayed.

Background Color -- the color against which text characters are displayed.

Text Color -- the colors to be used for displaying normal (i.e., unMarked) text.

Mark Color -- the colors to be used to indicate which character has been selected (i.e., Marked) in response to a MiniBASE M: or M/A: prompt. Foreground and Background colors may be set in the Mark: field independent of those set for the Text windows.

Allowable settings:

Setup will allow you to set any of the Window Color fields to one of the following values: White, Black, Blue, Green, Cyan, Red, Magenta, or Yellow. In addition, any color can be either Bold or Normal (selectable via the field immediately preceding each Foreground color).

Obviously some color combinations are better than others. First of all, if you have a monochrome display, you are probably better off sticking to settings of black and white (including the Bold settings); other color settings will work, but the effects will vary with the particular display and video board in use. Feel free to experiment; remember, you can always reset to the default values by using F5 in Setup.

Default Settings:

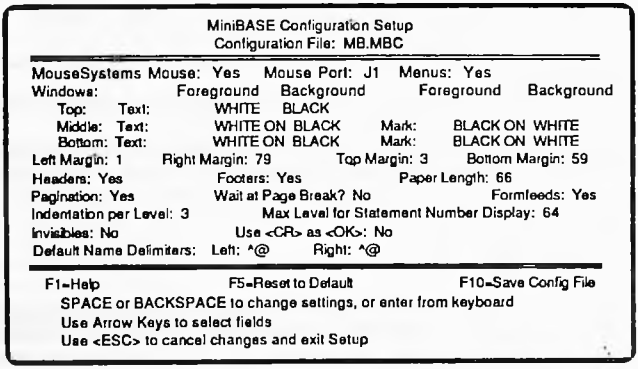
Windows:		Foreground	Background	Foreground	Background
Top:	Text:	WHITE	BLACK		
Middle:	Text: BOLD	WHITE	BLACK	Mark:	BLACK WHITE
Bottom:	Text:	WHITE	BLACK	Mark:	BLACK WHITE

Change via: Space Bar

Affects: Display only.

Z-100 Version

Setup for the Zenith Z-100 supports basically the same functions as the IBM & compatible version described above. Differences are illustrated in the figure below.



Window color settings other than black and white are not supported. The other differences, which relate primarily to support of the Mouse Systems mouse, are outlined below.

MouseSystems Mouse:

If you are using the Mouse Systems mouse with your Z-100, this value must be set to 'Yes' (the default) for it to function.

Allowable settings: Yes/No
Default Settings: Yes
Change via: Space Bar.
Affects: Display only.

Mouse Port:

Your Z-100 has two serial ports, J1 and J2. Your mouse (if you have one) must be plugged into one or the other, and MiniBASE needs to know where it is. This setting supplies that information.

Allowable settings: J1/J2
Default Settings: J1
Change via: Space Bar.
Affects: Display only.

Menus:

In order to use the popup menus that came on your Mouse Support disk, this setting must be 'Yes.'

Allowable settings: Yes/No
Default Settings: Yes
Change via: Space Bar.
Affects: Display only.

Changing Configuration Files: The Set Configuration Command

You might want to have several different configuration files. For example, a file named VISIBLE.MBC might be just like your default configuration file except for having the *Visibles* parameter set to Yes. Another file (DRAFT.MBC) might have printing parameters set for draft printing, while another (FINAL.MBC) was set to provide final printing.

To switch from one configuration file to another during a MiniBASE session, use the *Set Configuration* command. Type '<SP>secFILENAME<OK>', where FILENAME is the name of the configuration file you want to use. It is not necessary to enter the .MBC extension.

Please note that if you have used the Goto <OPT>Setup command to edit a configuration file during a MiniBASE session, you will have to use the Set Configuration command when you return to MiniBASE before the new settings will take effect.

COMMAND SUMMARY

Append Statement (at) MARK/ADDRESS (to statement at) MARK/ADDRESS (join with text) T/OK:

Break Statement (after Visible at) MARK LEVELADJUST OK:

Copy

Copy Branch (from) MARK/ADDRESS (to follow) MARK/ADDRESS LEVELADJUST OK:

Copy Character (from) MARK/ADDRESS (to follow) MARK/ADDRESS OK:

Copy Group (from) MARK/ADDRESS (thru) MARK/ADDRESS (to follow) MARK/ADDRESS LEVELADJUST OK:

Copy Invisible (from) MARK/ADDRESS (to follow) MARK/ADDRESS OK:

Copy Plex (from) MARK/ADDRESS (to follow) MARK/ADDRESS LEVELADJUST OK:

Copy Statement (from) MARK/ADDRESS (to follow) MARK/ADDRESS LEVELADJUST OK:

Copy Text (from) MARK/ADDRESS (thru) MARK/ADDRESS (to follow) MARK/ADDRESS OK:

Copy Visible (from) MARK/ADDRESS (to follow) MARK/ADDRESS OK:

Copy Word (from) MARK/ADDRESS (to follow) MARK/ADDRESS OK:

Delete

Delete Branch (at) MARK/ADDRESS OK OK:

Delete Character (at) MARK/ADDRESS OK OK:

Delete Group (at) MARK/ADDRESS (thru) MARK/ADDRESS OK OK:

Delete Invisible (at) MARK/ADDRESS OK OK:

Delete Plex (at) MARK/ADDRESS OK OK:

Delete Statement (at) MARK/ADDRESS OK OK:

Delete Text (at) MARK/ADDRESS (thru) MARK/ADDRESS OK OK:

Delete Visible (at) MARK/ADDRESS OK OK:

Delete Word (at) MARK/ADDRESS OK OK:

Execute

Goto

Goto Augterm OK:

Goto DOS OK:

Goto Programs OK:

Goto <OPT>PROGRAM OK:

Insert

Insert Branch (at) MARK/ADDRESS LEVELADJUST MARK/TYPEIN OK:

Insert Character (at) MARK/MARK/TYPEIN OK:

Insert Front (of statement at) MARK/ADDRESS OK:

Insert Group (at) MARK/ADDRESS LEVELADJUST MARK/TYPEIN OK:

Insert Invisible (at) MARK/MARK/TYPEIN OK:

Insert Plex (at) MARK/ADDRESS LEVELADJUST MARK/TYPEIN OK:

Insert Statement (at) MARK/ADDRESS LEVELADJUST MARK/TYPEIN OK:

Insert Text (at) MARK/MARK/TYPEIN OK:

Insert Visible (at) MARK/MARK/TYPEIN OK:

Insert Word (at) MARK/MARK/TYPEIN OK:

Jump

Jump (to) File TYPEIN VIEWSPECS OK:

Jump (to) <>Text File TYPEIN (using) One/Two/Paragraphing VIEWSPECS OK:

Jump (to) Origin (of file) MARK/ADDRESS VIEWSPECS OK:

Jump (to) Back MARK/ADDRESS VIEWSPECS OK:

Jump (to) <>Next MARK/ADDRESS VIEWSPECS OK:

Jump (to) Successor MARK/ADDRESS VIEWSPECS OK:

Jump (to) Up MARK/ADDRESS VIEWSPECS OK:

Jump (to) Down MARK/ADDRESS VIEWSPECS OK:

Jump (to) Predecessor MARK/ADDRESS VIEWSPECS OK:

Jump (to) Head MARK/ADDRESS VIEWSPECS OK:

Jump (to) Tail MARK/ADDRESS VIEWSPECS OK:

Jump (to) <>Character MARK/ADDRESS VIEWSPECS OK:

Jump (to) Content First MARK/TYPEIN VIEWSPECS OK:

Jump (to) Content Next MARK/TYPEIN VIEWSPECS OK:

Jump (to) Item MARK/ADDRESS VIEWSPECS OK:

Jump (to) MARK/ADDRESS VIEWSPECS OK:

Jump (to) Return OK:

Move

Move Branch (from) MARK/ADDRESS (to follow) MARK/ADDRESS LEVELADJUST OK:

Move Character (from) MARK/ADDRESS (to follow) MARK/ADDRESS OK:

Move Group (from) MARK/ADDRESS (thru) MARK/ADDRESS (to follow) MARK/ADDRESS LEVELADJUST OK:

Move Invisible (from) MARK/ADDRESS (to follow) MARK/ADDRESS OK:

Move Plex (from) MARK/ADDRESS (to follow) MARK/ADDRESS LEVELADJUST OK:

Move Statement (from) MARK/ADDRESS (to follow) MARK/ADDRESS LEVELADJUST OK:

Move Text (from) MARK/ADDRESS (thru) MARK/ADDRESS (to follow) MARK/ADDRESS OK:

Move Visible (from) MARK/ADDRESS (to follow) MARK/ADDRESS OK:

Move Word (from) MARK/ADDRESS (to follow) MARK/ADDRESS OK:

Quit

Replace

Replace All (in) STRUCTURE (at) MARK/ADDRESS (of old text) MARK/TYPEIN (by new text) MARK/TYPEIN OK:

Replace Branch (at) MARK/MARK/TYPEIN OK:

Replace Character (at) MARK/MARK/TYPEIN OK:

Replace Group (at) MARK/ADDRESS (thru) MARK/ADDRESS (by) MARK/TYPEIN OK:

Replace Invisible (at) MARK/MARK/TYPEIN OK:

Replace Plex (at) MARK/MARK/TYPEIN OK:

Replace Statement (at) MARK/MARK/TYPEIN OK:

Replace Text (at) MARK/ADDRESS (thru) MARK/ADDRESS (by) MARK/TYPEIN OK:

Replace Visible (at) MARK/MARK/TYPEIN OK:

Replace Word (at) MARK/MARK/TYPEIN OK:

<>Set Configuration (from file) TYPEIN OK:

<>Set Viewspecs VIEWSPECS OK:

<>Show Directory (for files) TYPEIN/OK OK:

Update

Update File OK:

Update File FILENAME OK:

Update Text File OK:

Update Text File FILENAME.TXT OK:

Index